МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії
(повна назва кафедри, циклової комісії)

КУРСОВА РОБОТА

з Основ програмування
(назва дисципліни)

на тему: Обернення матриці методами
окаймлення та розбиття на клітки

Студента 1 курсу, групи ІП-14
Медвідя Олександра Руслановича

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник Головченко Максим Миколайович
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: _____
Національна оцінка _____

Члени комісії

_____ ст. викладач Головченко М.М.
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ доцент Муха І.П.
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 2022 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрям "ІПЗ"

Курс 1 Група ІП-14 Семестр 2

ЗАВДАННЯ
на курсову роботу студента
Медвідя Олександра Руслановича

(прізвище, ім'я, по батькові)

1. Тема роботи Обернення матриці методами окаймлення та розбиття на клітки

2. Строк здачі студентом закінченої роботи 12.06.2022

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання 10.02.2022

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	10.02.2022	
2.	Підготовка ТЗ	02.05.2022	
3.	Пошук та вивчення літератури з питань курсової роботи	03.05.2022	
4.	Розробка сценарію роботи програми	04.05.2022	
6.	Узгодження сценарію роботи програми з керівником	04.05.2022	
5.	Розробка (вибір) алгоритму рішення задачі	04.05.2022	
6.	Узгодження алгоритму з керівником	04.05.2022	
7.	Узгодження з керівником інтерфейсу користувача	05.05.2022	
8.	Розробка програмного забезпечення	06.05.2022	
9.	Налагодження розрахункової частини програми	06.05.2022	
10.	Розробка та налагодження інтерфейсної частини програми	07.05.2022	
11.	Узгодження з керівником набору тестів для контрольного прикладу	25.05.2022	
12.	Тестування програми	26.05.2022	
13.	Підготовка пояснювальної записки	10.06.2022	
14.	Здача курсової роботи на перевірку	12.06.2022	
15.	Захист курсової роботи	18.06.2022	

Студент _____
(підпис)

Керівник _____ Головченко Максим Миколайович
(підпис) (прізвище, ім'я, по батькові)

"___" _____2022 р.

АНОТАЦІЯ

Пояснювальна записка до курсової: 148 сторінок, 20 рисунків, 14 таблиць, 3 посилання.

Об’єкт дослідження: Обернення матриці методами окаймлення та розбиття на клітки.

Мета роботи: дослідження методів обернення матриці окаймлення та розбиття на клітки, розробка програмного забезпечення для обернення квадратних, невироджених матриць методами окаймлення та розбиття на клітки.

Опановано розробку програмного забезпечення з використанням ООП. Приведені змістовні постановки задач, їх індивідуальні математичні моделі, а також описано детальний процес розв’язання кожної з них.

Виконана програмна реалізація обернення матриці методами окаймлення та розбиття на клітки.

МАТРИЦЯ, КВАДРАТНА МАТРИЦЯ, НЕВИРОДЖЕНА МАТРИЦЯ, ОБЕРНЕННЯ МАТРИЦІ, МЕТОД ОКАЙМЛЕННЯ, МЕТОД РОЗБИТТЯ НА КЛІТКИ.

ЗМІСТ

ВСТУП

Дана робота присвячена розробці програмного забезпечення для обернення квадратних, невироджених матриць методами окаймлення та розбиття на клітки з використанням об’єктно-орієнтованого програмування. Задача програмного забезпечення полягає в графічному та текстовому відображенні матриці, оберненої до вхідної та детального, покрокового опису рішення задачі заданими методами.

1 ПОСТАНОВКА ЗАДАЧІ

Розробити програмне забезпечення, що буде обертати матрицю даними методами:

а) Метод окаймлення;

б) Метод розбиття на клітки;

Вхідними даними є задана користувачем, або згенерована автоматично квадратна матриця. Програмне забезпечення повинно мати графічний інтерфейс, за допомогою якого можна вказувати значення матриці, задавати діапазон генерації значень, обирати метод обернення матриці та за бажанням користувача відображати детальне рішення задачі.

Вихідними даними для даної роботи є матриця, обернена до вхідної, що є розв’язком поставленої задачі, результат виводиться на екран та, за бажанням користувача, записується в окремий файл, ім’я якого вказує користувач.

2 ТЕОРИТИЧНІ ВІДОМОСТІ

Матриця – математичний об’єкт, записаний у вигляді прямокутної таблиці, він допускає операції віднімання, додавання, множення та множення на скаляр. Обернена матриця – матриця, яка при множенні на основну, дає одиничну матрицю, матрицю, з 1-ми на головній діагоналі та 0-ми на інших позиціях.

Обернену матрицю мають лише квадратні (кількість рядків та стовбців однакова) невироджені (визначник матриці не дорівнює 0).

1.1. Метод окаймлення

Метод окаймлення є ітеративним алгоритмом обернення матриці, який передбачає розгляд основної матриці по частинам та знаходження частин оберненої матриці за формулами. На першій ітерації розглядається матриця 1x1, тобто перший елемент основної матриці. На другій ітерації вже розглядається матриця 2x2, на третій 3x3 і т.д..

1.2. Метод розбиття на клітки

Метод розбиття на клітки є рекурсивним алгоритмом обернення матриці, який передбачає розбиття основної матриці на 4 блоки (клітки), потім знаходження 4их блоків вже оберненої матриці за формулами. Знайшовши оберненні клітки, треба зібрати їх так само, як розбиралась основна. Головною умовою при розбитті матриці на клітки є те, що клітка 11 та kk повинні бути квадратними.

3 ОПИС АЛГОРИТМІВ

3.1. Перелік всіх основних змінних та їхнє призначення наведено в таблиці

Таблиця 3.1 – Основні змінні та їхні призначення

Змінна	Призначення
MainMatrix	Основна матриця
ReverseMatrix	Обернена матриця
Disposal	Представлення розмірності матриці
OldA	Змінна яка використовується у методі окаймлення
A	Змінна яка використовується у методі окаймлення
k	Змінна яка використовується у методі окаймлення
U	Змінна яка використовується у методі окаймлення
V	Змінна яка використовується у методі окаймлення
a	Змінна яка використовується у методі окаймлення
akk	Змінна яка використовується у методі окаймлення
r	Змінна яка використовується у методі окаймлення
q	Змінна яка використовується у методі окаймлення
B	Змінна яка використовується у методі окаймлення
MatrixCell11	Клітка основної матриці

MatrixCell12	Клітка основної матриці
MatrixCell21	Клітка основної матриці
MatricxCell22	Клітка основної матриці
MatrixR11	Клітка оберненої матриці
MatrixR12	Клітка оберненої матриці
MatrixR21	Клітка оберненої матриці
MatrixR22	Клітка оберненої матриці
number	Змінна, яка показує на номер клітки матриці
type	Змінна, яка відповідає за обраний метод
ZeroCheck	Рядок, який відображає число матриці
ZeroPlus	Рядок, який відображає -0,000
ZeroMinus	Рядок, який відображає 0,000
temp	Визначник матриці
TempMatrix	Поточна матриця у функції викреслення рядка та стовбця
IndCol	Кількість стовбців матриці
IndRow	Кількість рядків матриці

3.2 Загальний алгоритм

1. ПОЧАТОК
2. Зчитати матрицю
3. ЯКЩО обраний метод окаймлення:
   3.1. ЯКЩО UnAcceptable(матриця, 1):
      3.1.1. ТО видати повідомлення про те, що матриця не підходить для заданого методу.

3.2. ІНАКШЕ знайти обернену матрицю методом окаймлення.

4. ЯКЩО обраний метод розбиття на клітки:

4.1. ЯКЩО UnAcceptable(матриця, 2):

4.1.1 ТО видати повідомлення про те, що матриця не підходить для заданого методу.

4.2. ІНАКШЕ знайти обернену матрицю методом розбиття на клітки.

5. Зчитати елемент видачі детального рішення

5.1. ЯКЩО обрана видача детального рішення

5.1.1 ТО видати детальне рішення згідно методу

5.2. ІНАКШЕ перейти до пункту 6

6. Вивести обернену матрицю

7. КІНЕЦЬ

3.3 Алгоритм методу окаймлення (ітеративний)

1. ПОЧАТОК
2. Disposal = розмірність введеної матриці
3. Розмірність матриці OldA – 1x1
4. ЯКЩО розмірність головної матриці дорівнює 1
    4.1. Повернути OldA та перейти до пункту 5
5. Цикл k = 0, 1, 2… поки k < Disposal
    5.1. Розмірність матриці A k+1
    5.2. Цикл i = 0, 1, 2… поки i < k+1
        5.2.1. Цикл j = 0, 1, 2… поки j < k+1
            5.2.1.1. Елемент [i][j] матриці A дорівнює елементу [i][j] матриці MainMatrix
    5.3. Розмірність матриці U – k, 1
    5.4. Розмірність матриці V – 1, k
    5.5. Число a = елемент [k][k] матриці A
    5.6. Цикл i = 0, 1, 2… поки i < k

5.6.1. Цикл j = 0, 1, 2… поки j < k

5.6.1.1. Елемент [i][0] матриці U дорівнює елементу [i][k] матриці A

5.6.1.2. Елемент [0][i] матриці V дорівнює елементу [k][i] матриці A

5.7. Число akk = a – (V * OldA * U) елемент [0][0]

5.8. Матриця r = OldA * U * (-1 / akk)

5.9. Матриця q = V * OldA * (-1 / akk)

5.10. Матриця B = OldA – (OldA * U) * q

5.11. Цикл i = 0, 1, 2… поки i < k

5.11.1. Цикл j = 0, 1, 2… поки j < k

5.11.1.1. Елемент [i][j] матриці A дорівнює елементу [i][j] матриці B

5.12. Цикл i = 0, 1, 2… поки i < k

5.12.1. Цикл j = 0, 1, 2… поки j < k

5.12.1.1. Елемент [j][k] матриці A дорівнює елементу [j][0] матриці r

5.12.1.2. Елемент [k][j] матриці A дорівнює елементу [0][j] матриці q

5.13. Елемент [k][k] дорівнює 1/akk

5.14. ЯКЩО k дорівнює Disposal – 1

5.14.1. ТО повернути матрицю A, відпрацьовану у функції MachineZero

5.15. OldA дорівнює A

6. КІНЕЦЬ

3.4. Опис алгоритму методу розбиття на клітки (рекурсивний)

1. ПОЧАТОК

2. ЯКЩО MainMatrix.Det() == 0:

2.1. (*ZeroDiv) = true

3. ЯКЩО розмірність MainMatrix дорівнює 1

3.1. MainMatrix.GetArr()[0][0] == 0:

3.1.1. (*ZeroDiv) = true

3.2. ТО елемент [0][0] матриці MainMatrix дорівнює 1 / елемент [0][0] матриці MainMatrix

3.3. Повернути MainMatrix

4. Матриця MatrixCell11 дорівнює CellDivisioninitialization(MainMatrix, 1)

5. Матриця MatrixCell12 дорівнює CellDivisioninitialization(MainMatrix, 2)

6. Матриця MatrixCell21 дорівнює CellDivisioninitialization(MainMatrix, 3)

7. Матриця MatrixCell22 дорівнює CellDivisioninitialization(MainMatrix, 4)

8. Матриця MatrixCell22Inverse = CellDivisioninitialization(MainMatrix, 1)

9. Матриця MatrixR11 = CellDivision(MatrixCell11 – (MatrixCell12 * (MatrixCell22Inverse * MatrixCell21)), count, ZeroDiv)

10. Матриця MatrixR12 = ((MatrixR11 * (-1)) * MatrixCell12) * MatrixCell22Inverse

11. Матриця MatrixR21 = (MatrixCell22Inverse * (-1)) * MatrixCell21 * MatrixR11

12. Матриця MatrixR22 = MatrixCell22Inverse – (MatrixCell22Inverse * MatrixCell21 * MatrixR12)

13. Матриця ReverseMatrix CellDivisionBuild(MainMatrix, MatrixCell11, MatrixCell12, MatrixCell21, MatrixCell22)

14. Повернути ReverseMatrix оброблену в MachineZero

15. КІНЕЦЬ

3.5. Опис функції CellDivisionInitialization(MainMatrix, number), яка розбирає матрицю на клітки

1. ПОЧАТОК
2. Disposal = розмірність введеної матриці
3. ЯКЩО Disposal є парним , ТО
    - 3.1. Розмірність MatrixCell11 – Disposal/2
    - 3.2. Розмірність MatrixCell12 – Disposal/2
    - 3.3. Розмірність MatrixCell21 – Disposal/2
    - 3.4. Розмірність MatrixCell22 – Disposal/2
    - 3.5. Цикл i = 0, 1, 2… поки i < Disposal
        - 3.5.1. Цикл j = 0, 1, 2… поки j < Disposal
            - 3.5.1.1. ЯКЩО i < Disposal / 2 та j < Disposal / 2
                - 3.5.1.1.1. ТО MatrixCell11 елемент [i][j] = MainMatrix елемент [i][j]
            - 3.5.1.2. ЯКЩО i < Disposal / 2 та j >= Disposal / 2
                - 3.5.1.2.1. ТО MatrixCell12 едемент [i][j – Disposal / 2] = MainMatrix елемент [i][j]
            - 3.5.1.3. ЯКЩО i >= Disposal / 2 та j < Disposal / 2
                - 3.5.1.3.1. ТО MatrixCell21 елемент [i – Disposal / 2][j] = MainMatrix елемент [i][j]
            - 3.5.1.4. ЯКЩО i >= Disposal / 2 та j >= Disposal / 2
                - 3.5.1.4.1. ТО MatrixCell22 елемент [i – Disposal / 2][j – Disposal / 2] = MainMatrix елемент [i][j]
    - 3.6. ЯКЩО number = 1
        - 3.6.1. ТО повернути MatrixCell11
    - 3.7. ЯКЩО number = 2
        - 3.7.1. ТО повернути MatrixCell12
    - 3.8. ЯКЩО number = 3
        - 3.8.1. ТО повернути MatrixCell21
    - 3.9. ЯКЩО number = 4
        - 3.9.1. ТО повернути MatrixCell22
4. ІНАКШЕ

4.1. Розмірність MatrixCell11 – Disposal / 2 + 1, Disposal / 2 + 1

4.2. Розмірність MatrixCell12 – Disposal / 2 + 1, Disposal / 2

4.3. Розмірність MatrixCell21 – Disposal / 2, Disposal / 2 + 1

4.4. Розмірність MatrixCell22 – Disposal / 2, Disposal / 2

4.5. Цикл i = 0, 1, 2… поки i < Disposal

4.5.1. Цикл j = 0, 1, 2… поки j < Disposal

4.5.1.1. ЯКЩО i < Disposal / 2 + 1 та j < Disposal / 2 + 1

4.5.1.1.1. ТО MatrixCell11 елемент [i][j] = MainMatrix елемент [i][j]

4.5.1.2. ЯКЩО i < Disposal / 2 + 1 та j >= Disposal / 2 + 1

4.5.1.2.1. ТО MatrixCell12 едемент [i][j – Disposal / 2 + 1] = MainMatrix елемент [i][j]

4.5.1.3. ЯКЩО i >= Disposal / 2 + 1 та j < Disposal / 2 + 1

4.5.1.3.1. ТО MatrixCell21 елемент [i – Disposal / 2 + 1][j] = MainMatrix елемент [i][j]

4.5.1.4. ЯКЩО i >= Disposal / 2 + 1 та j >= Disposal / 2 + 1

4.5.1.4.1. ТО MatrixCell22 елемент [i – Disposal / 2 + 1][j – Disposal / 2 + 1] = MainMatrix елемент [i][j]

4.6. ЯКЩО number = 1

4.6.1. ТО повернути MatrixCell11

4.7. ЯКЩО number = 2

4.7.1. ТО повернути MatrixCell12

4.8. ЯКЩО number = 3

4.8.1. ТО повернути MatrixCell21

4.9. ЯКЩО number = 4

4.9.1. ТО повернути MatrixCell22

5. КІНЕЦЬ

3.6. Опис функції CellDivisionBuild(MainMatrix, number), яка збирає клітки у матрицю

1. ПОЧАТОК

2. Розмірність матриці ReverseMatrix дорівнює розмірності матриці MainMatrix

3. Disposal дорівнює розмірності ReverseMatrix

4. ЯКЩО Disposal є парним

4.1 . Цикл i = 0, 1, 2… поки i < Disposal

4.1.1. Цикл j = 0, 1, 2… поки j < Disposal

4.1.1.1. ЯКЩО i < Disposal / 2 та j < Disposal / 2

4.1.1.1.1. ТО ReverseMatrix елемент [i][j] = MatrixR11 елемент [i][j]

4.1.1.2. ЯКЩО i < Disposal / 2 та j >= Disposal / 2

4.1.1.2.1. ТО ReverseMatrix елемент [i][j] = MatrixR12 елемент [i][j – Disposal / 2]

4.1.1.3. ЯКЩО i >= Disposal / 2 та j < Disposal / 2

4.1.1.3.1. ТО ReverseMatrix елемент [i][j] = MatrixR21 елемент [i – Disposal / 2][j]

4.1.1.4. ЯКЩО i >= Disposal / 2 та j >= Disposal / 2

4.1.1.4.1. ТО ReverseMatrix елемент [i][j] = MatrixR22 елемент [i – Disposal / 2][j – Disposal / 2]

5. ІНАКШЕ

5.1. Цикл i = 0, 1, 2… поки i < Disposal

5.1.1. Цикл j = 0, 1, 2… поки j < Disposal

5.1.1.1. ЯКЩО i < Disposal / 2 + 1 та j < Disposal / 2 + 1

5.1.1.1.1. ТО ReverseMatrix елемент [i][j] = MatrixR11 елемент [i][j]

5.1.1.2. ЯКЩО i < Disposal / 2 + 1 та j >= Disposal / 2 + 1

5.1.1.2.1. ТО ReverseMatrix елемент [i][j] = MatrixR12 елемент [i][j – (Disposal / 2 + 1)]

5.1.1.3. ЯКЩО i >= Disposal / 2 + 1 та j < Disposal / 2 + 1

5.1.1.3.1. ТО ReverseMatrix елемент [i][j] = MatrixR21 елемент [i – (Disposal / 2 + 1)][j]

5.1.1.4. ЯКЩО i >= Disposal / 2 + 1 та j >= Disposal / 2 + 1

5.1.1.4.1. ТО ReverseMatrix елемент [i][j] = MatrixR22 елемент [i – (Disposal / 2 + 1)][j – (Disposal / 2 + 1)]

6. Повернути ReverseMatrix

7. КІНЕЦЬ

3.7. Опис функції MachineZero(CurrentMatrix), яка перевіряє матрицю на машинні нулі

1. ПОЧАТОК

2. Цикл i = 0, 1, 2… поки i < Disposal

2.1. Цикл j = 0, 1, 2… поки j < Disposal

2.1.1. ЯКЩО DoubleZero(CurrentMatrix елемент [i][j])

2.1.1.1. ТО елемент [i][j] матриці CurrentMatrix дорівнює 0

3. Повернути CurrentMatrix

4. КІНЕЦЬ

3.8. Опис функції DoubleZero(number), яка перевіряє число на машинні нулі

1. ПОЧАТОК

2. Рядок ZeroCheck дорівнює переведений у рядок число number

3. Рядок ZeroMinus дорівнює “-0,000”

4. Рядок ZeroPlus дорівнює “0,000”

5. ЯКЩО ZeroCheck дорівнює ZeroPlus

5.1. Повернути true

6. ЯКЩО ZeroCheck дорівнює ZeroMinus

6.1. Повернути true

7. ІНАКШЕ

7.1. Повернути false

8. КІНЕЦЬ

3.9. Опис функції UnAcceptable(MainMatrix, type), яка перевіряє число на машинні нулі

1. ПОЧАТОК

2. ЯКЩО матриця MainMatrix розмірності 1 ТА перший елемент не дорівнює 0

2.1. ТО повернути false

3. ЯКЩО MainMatrix.Det() дорівнює 0

3.1. ТО повернути true

4. ЯКЩО type дорівнює 1 АБО 3, ТО

4.1. ЯКЩО перший елемент MainMatrix дорівнює 0

4.1.1. ТО повернути true

4.2. Цикл k = 0, 1, 2… поки k < розмірність MainMatrix

4.2.1. Розмірність матриці Temp дорівнює k

4.2.2. Цикл i = 0, 1, 2… поки i < k

4.2.2.1. Цикл j = 0, 1, 2… поки j < k

4.2.2.1.1. Елемент [i][j] матриці Temp дорівнює елементу [i][j] матриці MainMatrix

4.2.3. ЯКЩО Temp.Det() дорівнює 0

4.2.3.1. ТО Повернути true

5. ЯКЩО type дорівнює 2 АБО 3:

5.1. count = 0

5.2. ZeroDiv = false

5.3. CellDivision(MainMatrix, &count, & ZeroDiv)

5.4. ЯКЩО ZeroDiv:

5.4.1. Повернути true

6. КІНЕЦЬ

3.10. Опис функції Det(), яка повертає визначник матриці і яка є методом класу Matrix

1. ПОЧАТОК
2. Temp дорівнює 0
3. k = 1
4. розмірність CurrentMatrix дорівнює розмірності поточної матриці
5. ЯКЩО розмірність поточної матриці дорівнює 1
   5.1. ТО temp першому елементу поточної матриці
6. ЯКЩО розмірність поточної матриці дорівнює 2
   6.1. ТО temp = [0][0] * [1][1] – [1][0] * [0][1] поточної матриці
7. ІНАКШЕ
   7.1. Цикл i = 0, 1, 2… i < розмірність
      7.1.1. m дорівнює розмірність – 1
      7.1.2. Розмірність TempMatrix – m
      7.1.3. TempMatrix.GetMatr(CurrentMatrix, 0, i)
      7.1.4. temp = temp + k * Arr[0][i] * TempMatrix.Det()
      7.1.5. k = -k
8. Повернути temp
9. КІНЕЦЬ

3.11. Опис функції GetMatr(CurrentMatrix, IndRow, IndCol), яка викреслює рядок та стовбець яка є методом класу Matrix

1. ПОЧАТОК
2. ki дорівнює 0

3. Цикл i = 0, 1, 2…, поки i < розмірність матриці

3.1. ЯКЩО i не дорівнює IndCol

3.1.1. Цикл j = 0, 1, 2…, поки j < розмірність матриці

3.1.1.1. ЯКЩО j не дорівнює IndCol

3.1.1.1.1. Елемент [ki][kj] поточної матриці дорівнює елементу [i][j] CurrentMatrix

4. КІНЕЦЬ

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Діаграма класів програмного забезпечення

Діаграма класів розробленого програмного забезпечення наведена на рисунку 4.1.

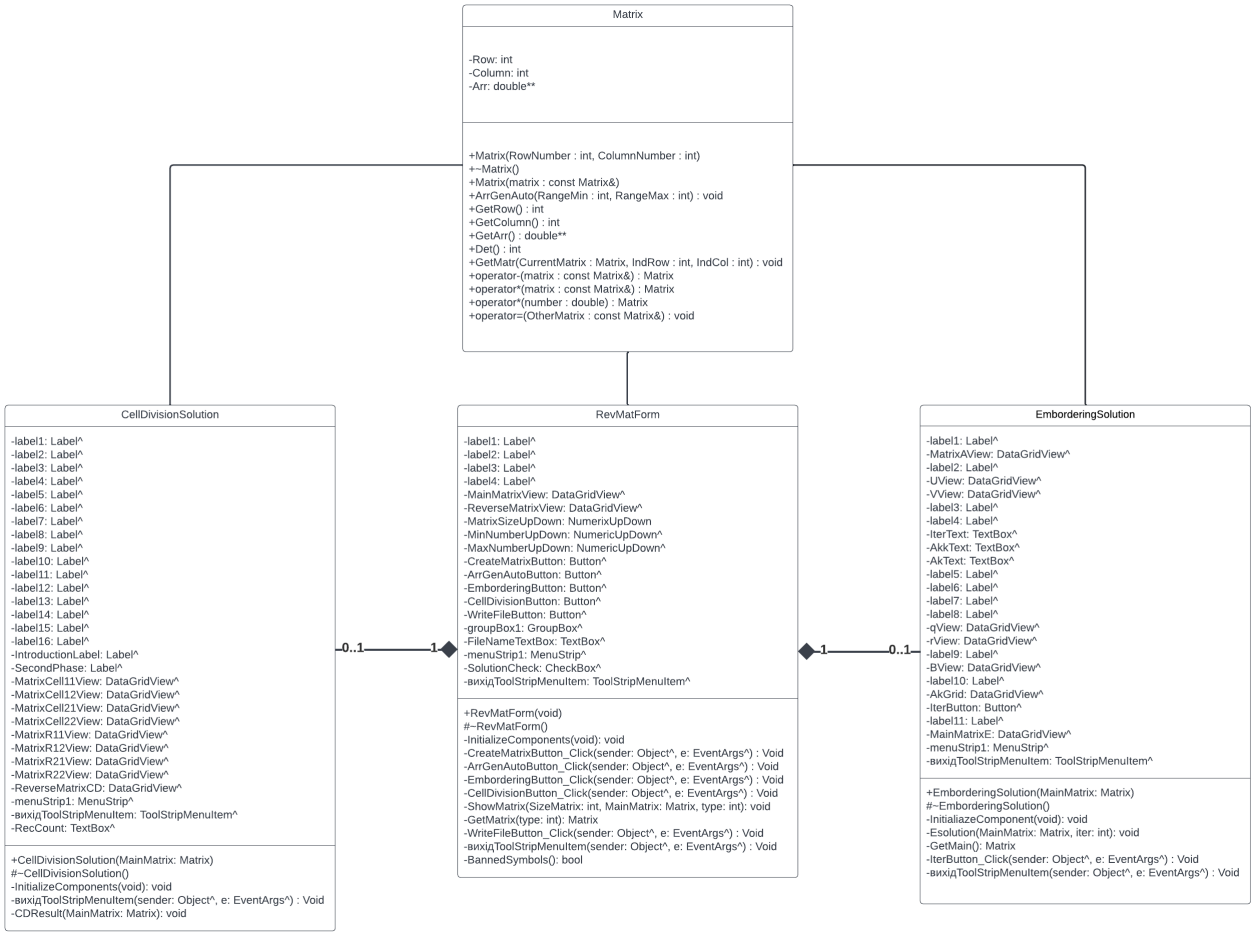


Рисунок 4.1 – Діаграма класів

4.2 Опис методів частин програмного забезпечення

4.2.1 Стандартні методи

У таблиці 4.1 наведено стандартні методи, використані при розробці програмного забезпечення.

Таблиця 4.1 – Стандартні методи

№ п/п	Назва классу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
1	string	find()	Returns an iterator to the first element in the range [first,last) tha t compares equal to *val*. If no such element is found, the function returns *last*.	-	iterator, int	string
2	string	length()	Returns the length of the string, in terms of bytes	-	Size, const	string
3	string	to_string()	Returns a string with the representation of *val*	val, int, double, long, float…	val, string	string
4	fstream	open()	Opens the file identified by argument *filename*, associating it with the stream object, so that input/output operations are performed on its content.	filename, const char*	void	fstream
5	fstream	close()	Closes the file currently associated with the object, disassociating it from the stream	-	void	fstream
6	cstdlib	rand()	Returns a pseudo-random integral number in the range between 0 and Ran d_Max	-	a sequence of apparently non-related numbers each time it is called, void	cstdlib
7	cstdlib	srand()	The pseudo-random number generator is initialized using the argument passed as *seed*	Seed, unsigned int	-	cstdlib
8	Convert	ToInt32()	Converts the specified string representation of a number to an equivalent 32-bit signed integer	value, string, object	value, int	-
9	Convert	ToDouble()	Converts the value of the specified double-precision floating-point number to an equivalent 32-bit signed integer	value, string, object	value, double	-

10	Convert	ToString()	Converts the specified value to its equivalent string representation.	value, int, double, float object	value, string	-
11	MessageBox	Show()	Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information. The message box returns an integer value that indicates which button the user clicked.	value, string	-	-
12	msclr/marshal_cppstd	marshal_as()	Converts data between native and managed environments	The value that you want to marshal to a To_Type variable	A variable of type To_Type that is the converted value of input	-
13	Application	Exit()	Breaks the sequence of the application	-	-	-
14	DatGridView	AutoResizeColumns()	Resizes Columns	-	-	-
15	DataGridView	AutoResizeRowHeadersWidth()	Resizes the width of row headers	-	-	-

4.2.2 Користувацькі методи

У таблиці 4.2 наведено користувацькі методи, створені при розробці програмного забезпечення.

Таблиця 4.2 – користувацькі методи

№ п/п	Назва классу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
1	Matrix	Matrix()	Конструктор класу Matrix	RowNumber, int, ColumnNumber, int	Matrix	Matrix.h
2	Matrix	Matrix()	Конструктор копіювання	matrix, const Matrix&	matrix, Matrix	Matrix.h
3	Matrix	~Matrix()	Деструктор класу Matrix	-	-	Matrix.h
4	Matrix	ArrGenAuto()	Генерація випадковим чисел матриці	RangeMin, int, RangeMax, int	-	Matrix.h
5	Matrix	GetRow()	Гетер кількості рядків матриці	-	Row, int	Matrix.h
6	Matrix	GetColumn	Гетер кількості	-	Column, int	Matrix.h

			стовбців матриці			
7	Matrix	GetArr()	Гетер двомірного масиву матриці	-	Arr, double**	Matrix.h
8	Matrix	Det()	Функція знаходження визначника матриці	-	Det, int	Matrix.h
9	Matrix	GetMatr()	Функція викреслення рядка та стовбця, для знаходження визначника	CurrentMatrix, Matrix, IndRow, int, IndCol, int	-	Matrix.h
10	Matrix	operator – ()	Перевантаження оператору – для віднімання матриць	matrix, const Matrix&	matrix, Matrix	Matrix.h
11	Matrix	operator * ()	Перевантаження * для множення матриць	matrix, const Matrix&	matrix, Matrix	Matrix.h
12	Matrix	operator * ()	Перевантаження * для множення матриці на число	number, double	matrix, Matrix	Matrix.h
13	Matrix	operator = ()	Перевантаження оператора присвоєння	OtherMatrix, const Matrix&	matrix, Matrix	Matrix.h
14	CellDivisionSolution	вихідToolStripMenuItem_Click	Подія виходу з програми	sender, Object^, e, EventArgs^	-	CellDivisionSolution.h
15	CellDivisionSolution	CDResult	Вивід результату методу розбиття на клітки	MainMatrix, Matrix	-	CellDivisionSolution.h
16	EmborderingSolution	Esolution	Вивід результату методу окаймлення	MainMatrix, Matrix, iter, int	-	EmborderingSolution.h
17	EmborderingSolution	GetMain	Отримання головної матриці для методу окаймлення	-	MainMatrix, Matrix	EmborderingSolution.h
18	EmborderingSolution	IterButton_Click	Зміна ітерацій у детальному рішенні метода окайлення	sender, Object^, e, EventArgs^	-	EmborderingSolution.h
19	EmborderingSolution	вихідToolStripMenuItem_Click	Подія виходу з програми	sender, Object^, e,	-	EmborderingSolution.h

				EventArgs^		
20	RevMat	DoubleZero	Перевірка елементу матриці на машинні нулі	number, double	true або false	RevMat.h
21	RevMat	UnAcceptable	Перевірка матриці на можливість використання у методі	MainMatrix, Matrix, type, int	true або false	RevMat.h
22	RevMat	MachineZero	Перевірка матриці на машинні нулі	CurrentMatrix, Matrix	CurrentMatrix, Matrix	RevMat.h
23	RevMat	Embordering	Обернення матриці методом окаймлення	MainMatrix, Matrix	ReverseMatrix, Matrix	RevMat.h
24	RevMat	CellDivisionInitialization	Розбиття матриці на клітки	MainMatrix, Matrix, number, int	MatrixCell11, Matrix	RevMat.h
25	RevMat	CellDivisionBuild	Зібрання кліток у матрицю	MainMatrix, MatrixR11, MatrixR12, MatrixR21, MatrixR22, Matrix	ReverseMatrix, Matrix	RevMat.h
26	RevMat	CellDivision	Обернення матриці методом розбиття на клітки	MainMatrix, Matrix, count, int*	ReverseMatrix, Matrix	RevMat.h
27	RevMat	WriteFile	Функція запису матриці у файл	ReverseMatri, Matrix, FileName, std::string	number, int	RevMat.h
28	RevMatForm	CreateMatrixButton_Click	Подія створення матриці	sender, Object^, e, EventArgs	-	RevMatForm.h
29	RevMatForm	ArrGenAutoButton_Click	Подія генерації матриці випадковим чином	sender, Object^, e, EventArgs	-	RevMatForm.h
30	RevMatForm	EmborderingButton_Click	Подія рішення задачі методом окаймлення	sender, Object^, e, EventArgs	-	RevMatForm.h
31	RevMatForm	CellDivisionButton_Click	Подія рішення задачі методом розбиття на клітки	sender, Object^, e, EventArgs	-	RevMatForm.h
32	RevMatForm	ShowMatrix	Функція відображення матриці	SizeMatrix, int, MainMatrix, Matrix, type, int	-	RevMatForm.h
33	RevMatForm	GetMatrix	Функція зчитування матриці	type, int	MainMatrix, Matrix	RevMatForm.h

34	RevMatForm	WriteFileButton_Click	Подіяязапис у матриці у файл	sender, Object^, e, EventArgs	-	RevMatForm.h
35	RevMatForm	вихідToolStripMenuItem_Click	Вихід з програми	sender, Object^, e, EventArgs	-	RevMatForm.h
36	RevMatForm	BannedSymbols	Функція перевірки матриці на некоректні символи		true або false	RevMatForm.h

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 План тестування

Складемо план тестування програмного забезпечення, за допомогою якого протестуємо весь основний функціонал та реакцію на виключні ситуації

а) Тестування правильності значень у вихідному файлі

б) Тестування правильності результатів рішення задач.

1) Тестування правильності результатів методу окаймлення.

2) Тестування правильності результатів методу розбиття на клітки.

в) Тестування програми при введені некоректних символів у матрицю

г) Тестування програми при введені виродженої матриці

д) Тестування програми при некоректному діапазоні генерації чисел

1) Тестування, коли мінімальний діапазон більший за максимальний.

2) Тестування, коли мінімальний та максимальний діапазон збігаються.

е) Тестування програми при непідходящих до методу матриць

1) Тестування, коли в одній з ітерації методу окаймлення, елемент має визначник 0.

2) Тестування, коли в методі розбиття на клітки при обертанні матриці в одній з рекурсій виходить ділення на 0.

5.2 Приклади тестування

Проведемо тестування програмного забезпечення згідно з розробленим планом, фіксуючи мету, початковий стан програми, вхідні дані, схему проведення, очікуваний результат і стан програми після проведення випробувань кожного тесту в окрему таблицю (таблиці 5.1-5.14).

Таблиця 5.1 - Тестування правильності значень у вихідному файлі

Мета тесту	Тестування правильності значень у вихідному файлі
Початковий стан програми	Згенерована матриця випадковим чином та знайдена обернена до неї одним з методів.
Вхідні дані	Обернена матриця 4x4
Схема проведення тесту	Вписати назву файлу та перевірити правильність записаних значень у ньому
Очікуваний результат	У вказаному файлі відображена розмірність та значення оберненої матриці
Стан програми після проведення випробувань	У вказаному файлі відображена розмірність та значення оберненої матриці

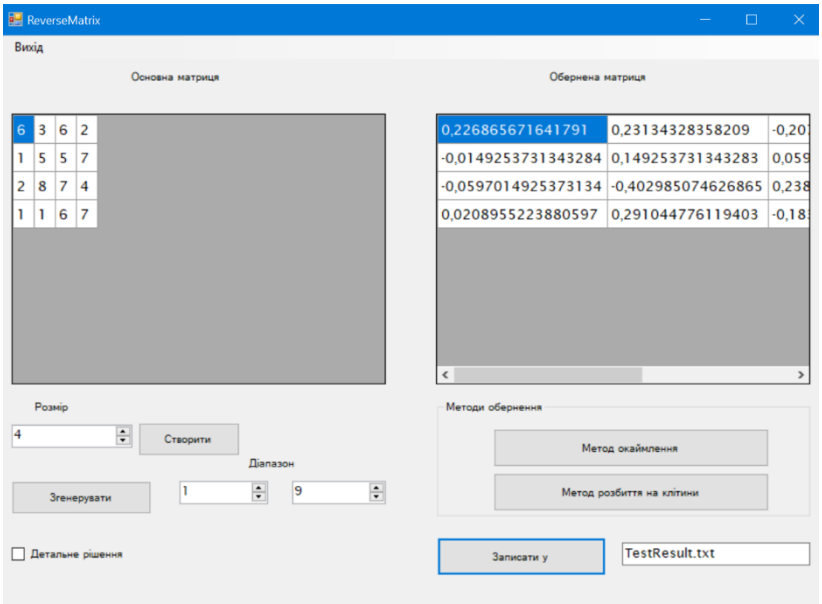


Рисунок 5.1 – Запис у файл

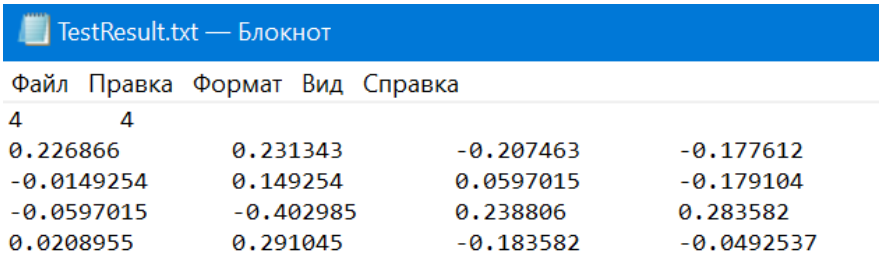


Рисунок 5.2 – Результати, записані у файл

Таблиця 5.2 - Тестування правильності результатів методу окаймлення.

Мета тесту	Тестування правильності результатів методу окаймлення
Початковий стан програми	Згенерована матриця випадковим чином
Вхідні дані	Матриця 4x4
Схема проведення тесту	Знайти обернену матрицю та перевірити значення у мат. калькуляторі
Очікуваний результат	Правильна обернена матриця
Стан програми після проведення випробувань	Правильна обернена матриця

Таблиця 5.3 - Тестування правильності результатів методу розбиття на клітки.

Мета тесту	Тестування правильності результатів методу розбиття на клітки
Початковий стан програми	Згенерована матриця випадковим чином
Вхідні дані	Матриця 4x4
Схема проведення тесту	Знайти обернену матрицю та перевірити значення у мат.

	калькуляторі
Очікуваний результат	Правильна обернена матриця
Стан програми після проведення випробувань	Правильна обернена матриця

Таблиця 5.4 - Тестування програми при введені виродженої матриці.

Мета тесту	Тестування програми при введені виродженої матриці
Початковий стан програми	Введена матриця з визначником 0
Вхідні дані	Матриця з визначником 0
Схема проведення тесту	Перевірити, як реагує програма на вироджену матрицю
Очікуваний результат	Повідомлення про недопустимість матриці у заданому методі
Стан програми після проведення випробувань	Повідомлення про недопустимість матриці у заданому методі

Таблиця 5.5 - Тестування, коли мінімальний діапазон більший за максимальний.

Мета тесту	Тестування, коли мінімальний діапазон більший за максимальний
Початковий стан програми	Мінімальний діапазон більший за максимальний

Вхідні дані	-
Схема проведення тесту	Перевірити, як реагує програма на некоректність діапазону
Очікуваний результат	Повідомлення про некоректність діапазону
Стан програми після проведення випробувань	Повідомлення про некоректність діапазону

Таблиця 5.6 - Тестування, коли мінімальний та максимальний діапазон збігаються.

Мета тесту	Тестування, коли мінімальний та максимальний діапазон збігаються
Початковий стан програми	Мінімальний та максимальний діапазон збігаються
Вхідні дані	-
Схема проведення тесту	Перевірити, як реагує програма на некоректність діапазону
Очікуваний результат	Повідомлення про некоректність діапазону
Стан програми після проведення випробувань	Повідомлення про некоректність діапазону

Таблиця 5.7 - Тестування програми при введені некоректних символів у матрицю.

Мета тесту	Тестування програми при введені некоректних символів у матрицю
Початковий стан програми	Матриця, з введеними словами
Вхідні дані	Матриця 4x4
Схема проведення тесту	Перевірити, як реагує програма на некоректну матрицю
Очікуваний результат	Повідомлення про некоректність матриці
Стан програми після проведення випробувань	Повідомлення про некоректність матриці

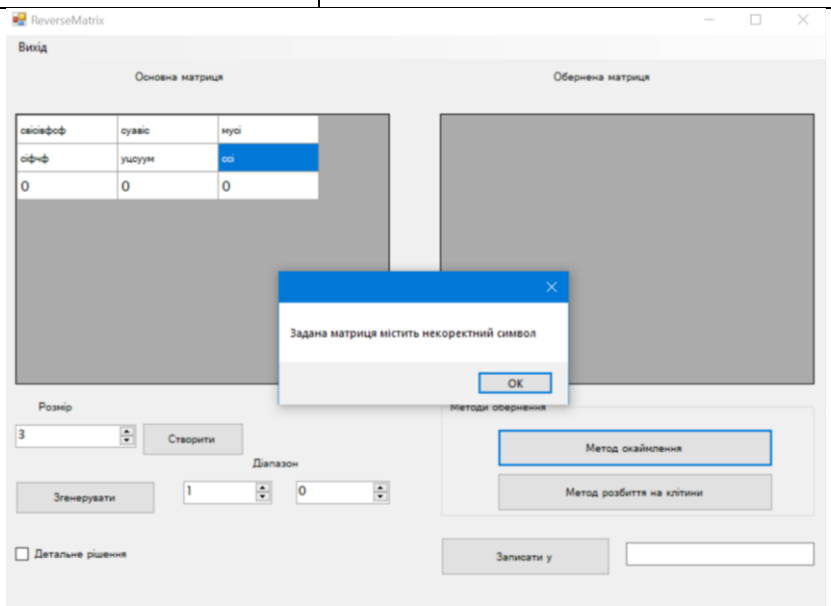


Рисунок 5.3 – Повідомлення про некоректність

Таблиця 5.8 - Тестування, коли в одній з ітерації методу окаймлення, елемент має визначник 0.

Мета тесту	Тестування, коли в одній з ітерації методу окаймлення, елемент має визначник 0
Початковий стан програми	Матриця, не підходяща для методу окаймлення
Вхідні дані	Матриця 4x4
Схема проведення тесту	Перевірити, як реагує програма на некоректну матрицю
Очікуваний результат	Повідомлення про некоректність матриці
Стан програми після проведення випробувань	Повідомлення про некоректність матриці

Таблиця 5.9 - Тестування, коли в методі розбиття на клітки при обертанні матриці в одній з рекурсій виходить ділення на 0.

Мета тесту	Тестування, коли в методі розбиття на клітки при обертанні матриці в одній з рекурсій виходить ділення на 0
Початковий стан програми	Матриця, не підходяща для методу розбиття на клітки
Вхідні дані	Матриця 4x4
Схема проведення тесту	Перевірити, як реагує програма на некоректну матрицю
Очікуваний результат	Повідомлення про некоректність матриці
Стан програми після проведення випробувань	Повідомлення про некоректність матриці

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1 Мануал

Після запуску виконавчого файлу з розширенням *.exe, відкривається головне вікно програми (Рисунок 6.1).

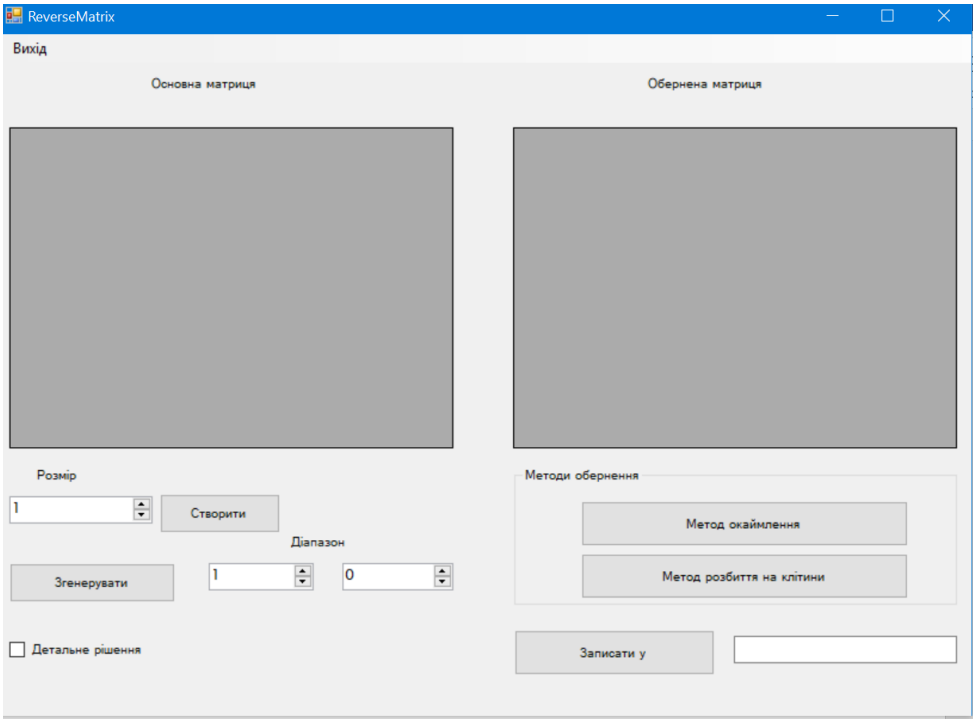


Рисунок 6.1 – Головне вікно програми

Далі натискаємо лівою кнопкою миші на елемент на стрілки під назвою “Розмір” та задаємо розмірність матриці, стрілка вверх збільшує розмірність, стрілка вниз – зменшує. Обравши розмірність матриці – натискаємо лівою кнопкою миші на кнопку “Створити”. Після цього створена матриця відображається на екрані. (Рисунок 6.2)

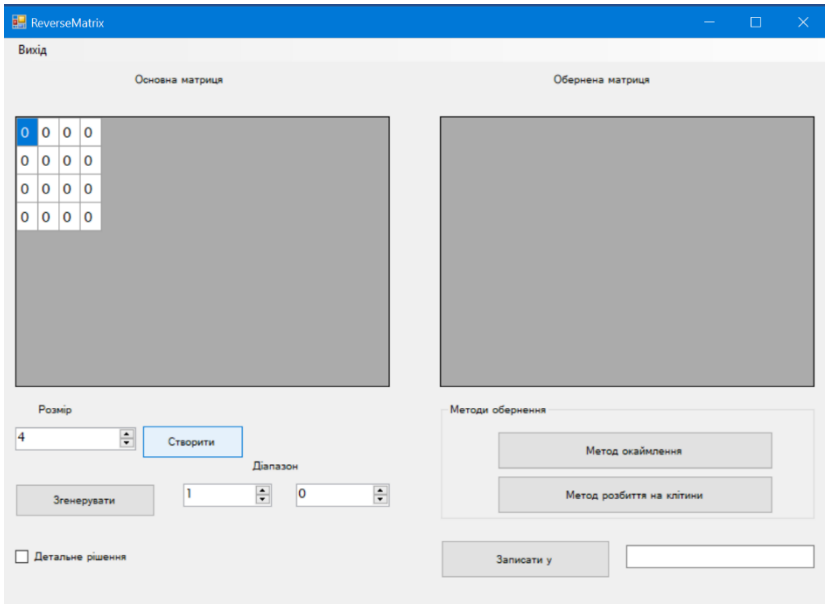


Рисунок 6.2 – Створена матриця

Далі є вибір: згенерувати матрицю випадково, або самостійно заповнити елементи матриці. Для генерації випадковим чином, задаємо діапазон генерації чисел за допомогою двох елементів під словом “Діапазон”. Після цього натискаємо лівою кнопкою миші на кнопку “Згенерувати”. Для самостійного заповнення матриці двічі натискаємо лівою кнопкою миші на комірку матриці та вводимо необхідне число за допомогою клавіатури. (Рисунок 6.3)

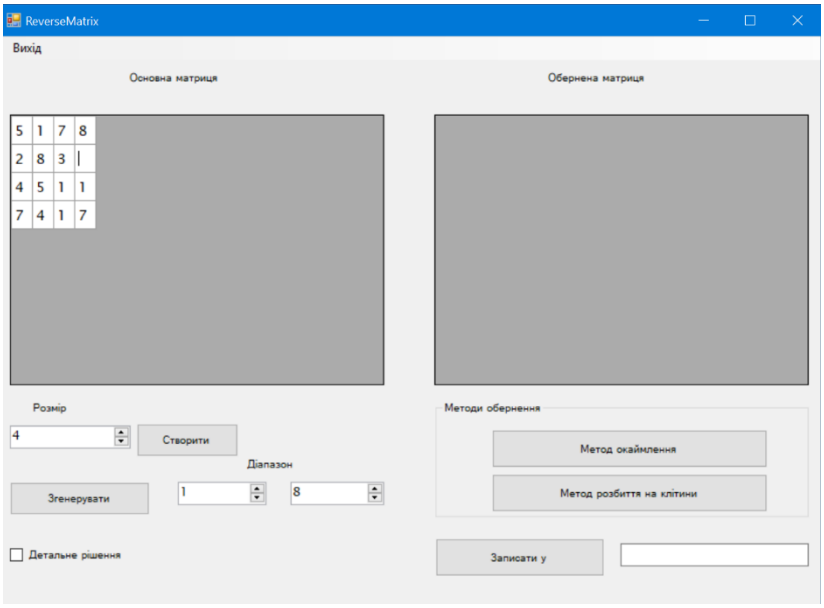


Рисунок 6.3 – заповнення матриці

Далі вибираємо метод обернення матриці, для цього натискаємо на одну з двох кнопок, розташованих у коробці “Методи обернення”. Натиснувши, зверху відобразиться обернена матриця (Рисунок 6.4)

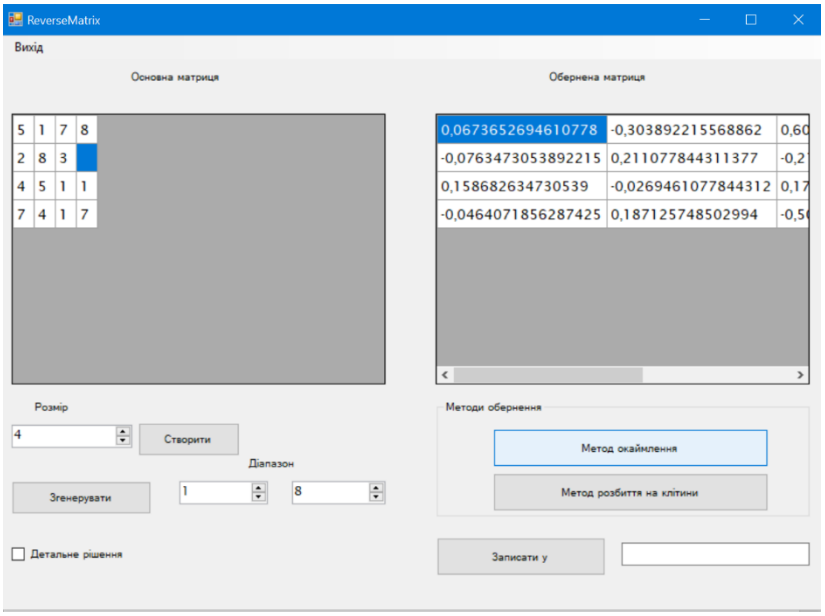


Рисунок 6.4 – відображення оберненої матриці

Для відображення детального рішення, натискаємо на елемент, розташований біля напису “Детальне рішення” (Рисунок 6.5). Після повторного натискання кнопок методів відобразиться вікна, які опишуть детально рішення методу окаймлення (Рисунок 6.6) та методу розбиття на клітини (Рисунок 6.7) відповідно.

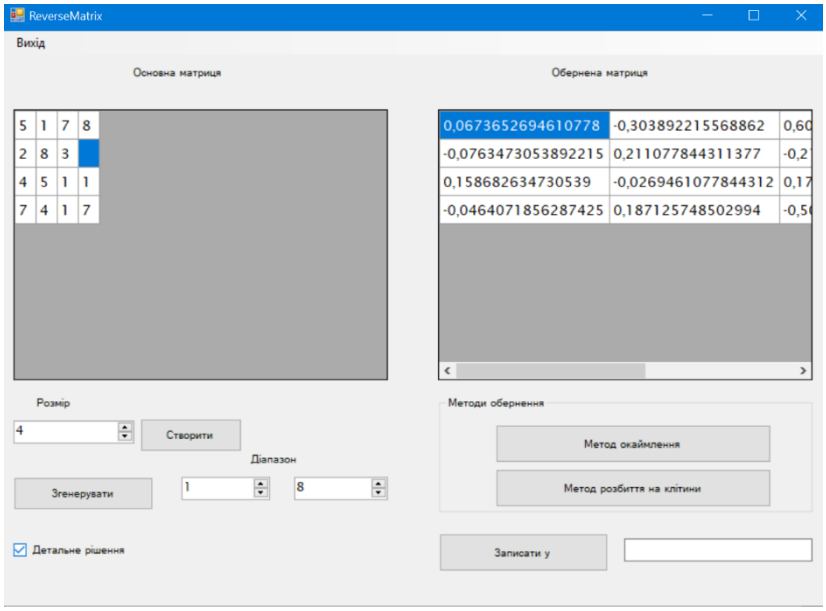


Рисунок 6.5 – обрання детального рішення

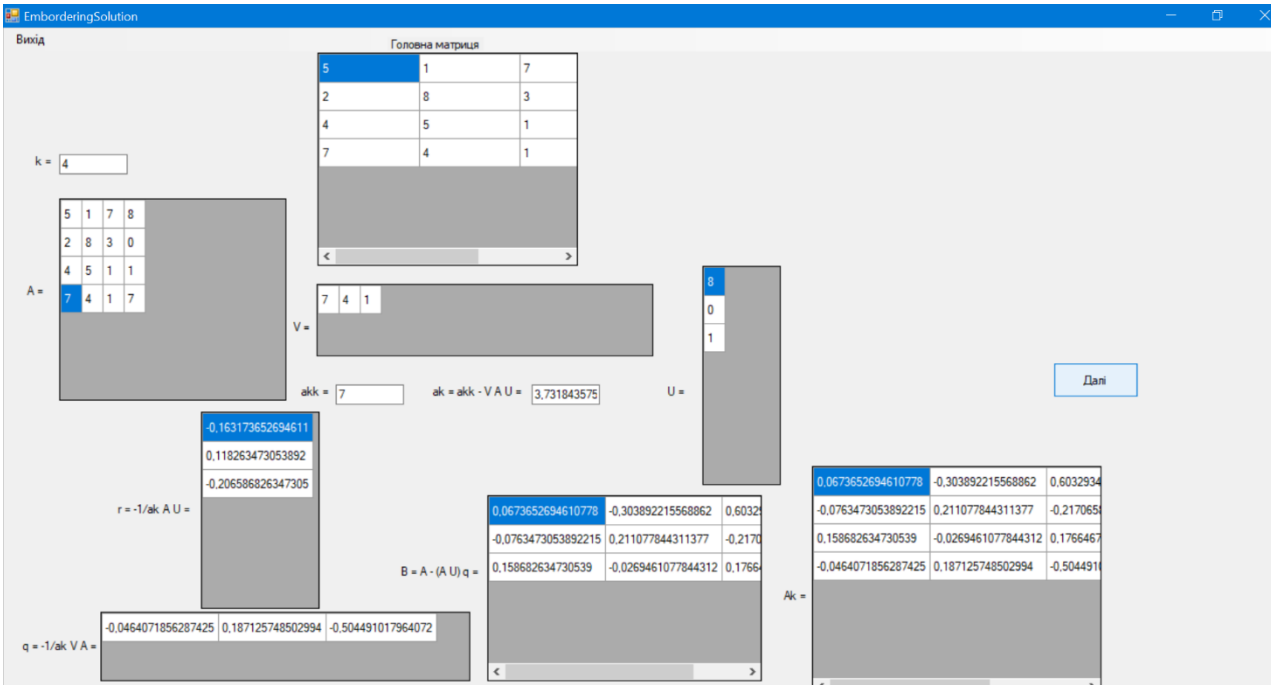


Рисунок 6.6 – детальне рішення методу окаймлення

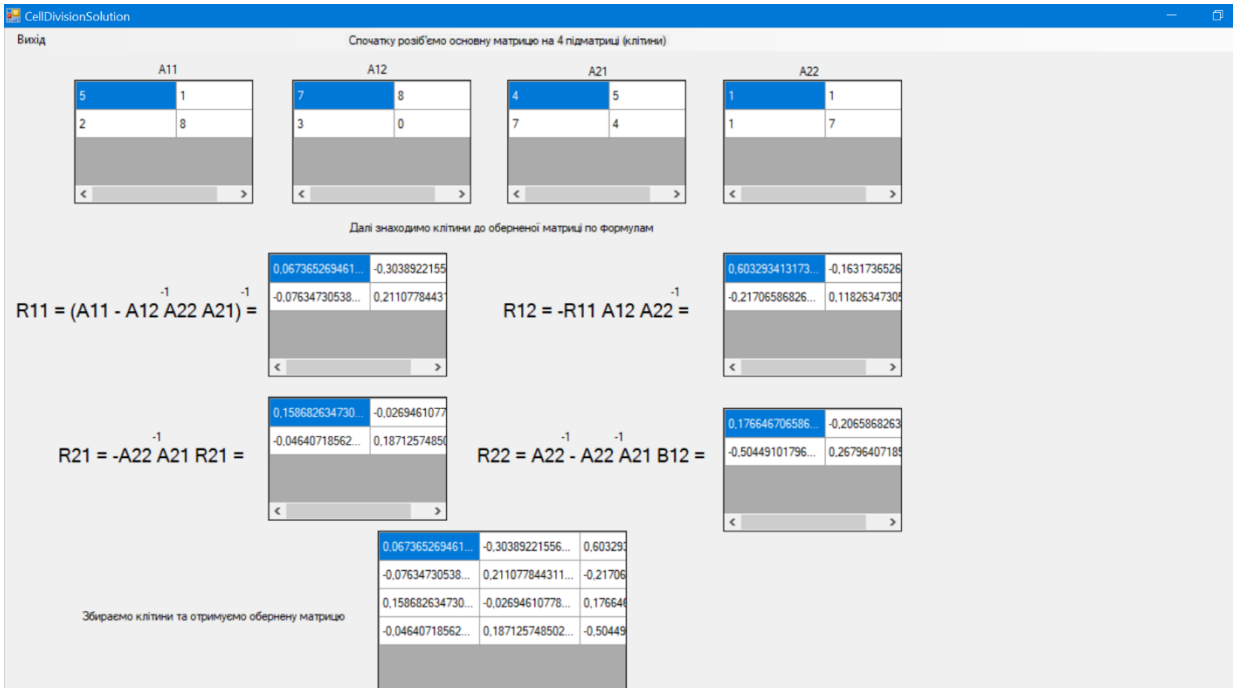


Рисунок 6.7 – детальне рішення методу розбиття на клітини

У детальному рішенні методу окаймлення натискаємо на кнопку “Далі” кожного разу коли хочемо переглянути нову ітерацію.

Для запису результату до файла, пишемо назву бажаного файла у комірку справа від кнопки “Записати у”. Після цього натискаємо лівою кнопкою миші на

вище згадану кнопку. Якщо файл був знайдений, то видається повідомлення про успішний запис. (Рисунок 6.8)

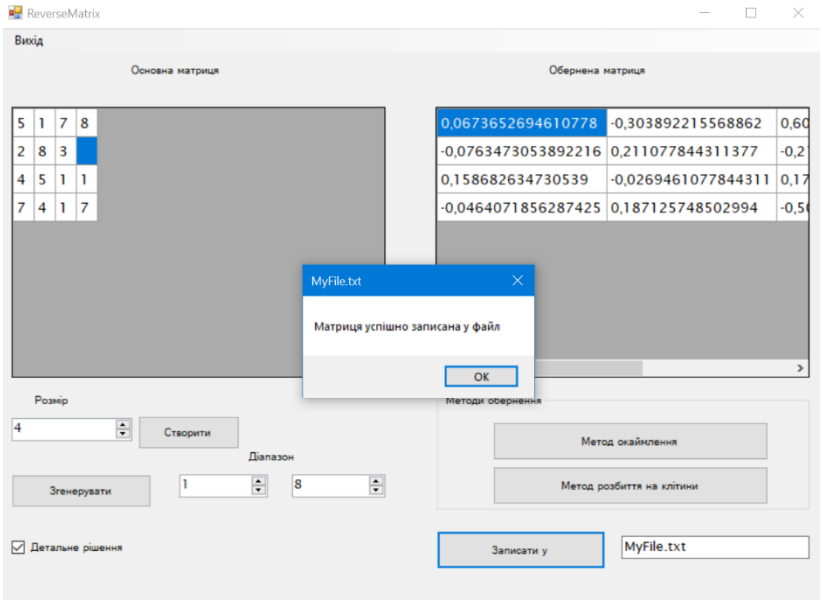


Рисунок 6.8 – Успішний запис до файлу.

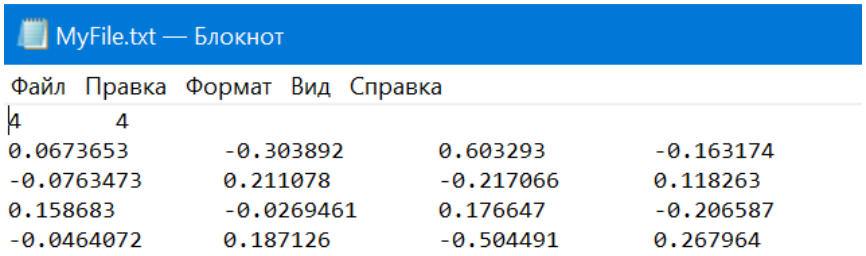


Рисунок 6.9 – Записана у файл обернена матриця

6.2. Формат вхідних та вихідних даних

Користувачем на вхід програми подається матриця заданої розмірності та з заданими, або згенерованими елементами.

Результатом виконання програми є матриця, обернена до вхідної обраним методом: окаймлення, або розбиттям на клітини.

6.3 Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1.

Таблиця 6.1 – Системні вимоги програмного забезпечення

	Мінімальні	Рекомендовані
Операційна система	Windows® XP/Windows Vista/Windows 7/ Windows 8/Windows 10/ Windows 11 (з останніми обновленнями)	Windows 7/ Windows 8/Windows 10/ Windows 11 (з останніми обновленнями)
Процесор	Intel® Pentium® III 1.0 GHz або AMD Athlon™ 1.0 GHz	Intel® Pentium® D або AMD Athlon™ 64 X2
Оперативна пам'ять	1 GB RAM (для Windows® XP/ Windows Vista/Windows 7/ Windows 8/Windows 10) / 4 GB RAM (для Windows 11)	4 GB RAM
Відеоадаптер	Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог)	
Дисплей	800x600	1024x768 або краще

Продовження таблиці 6.1

	Мінімальні	Рекомендовані
Прилади введення	Комп’ютерна миша	
Додаткове програмне забезпечення	Microsoft .Net Framework 4.7.2 або вище	

7 АНАЛІЗ І УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ

Головною задачею курсової роботи була реалізація програми для обернення матриці наступними методами: окаймлення, розбиття на клітки.

Критичні ситуації у роботі програми виявлені не були. Під час тестування було виявлено, що більшість помилок виникало тоді, коли деякі елементи матриці при методі окаймлення були виродженими та коли при одній з рекурсій методу розбиття на клітки в одній з кліток було ділення на нуль. Тому всі дані, які вводить користувач, або генеруються автоматично, перевіряються на валідність і лише потім подаються на обробку програмі.

Для перевірки та доведення достовірності результатів виконання програмного забезпечення скористаюся MS Excel:

а) Метод окаймлення.

Результат виконання методу окаймлення наведено на рисунку 7.1

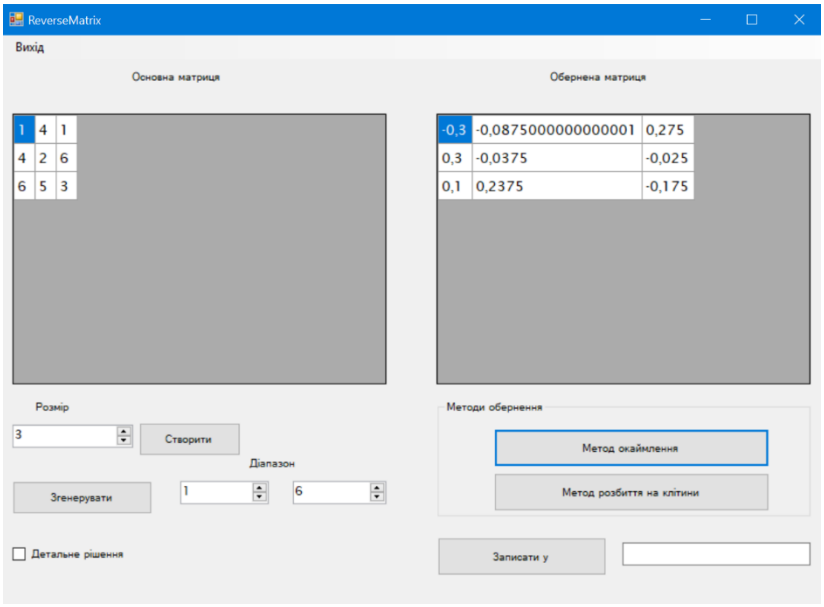


Рисунок 7.1 – Результат методу окаймлення

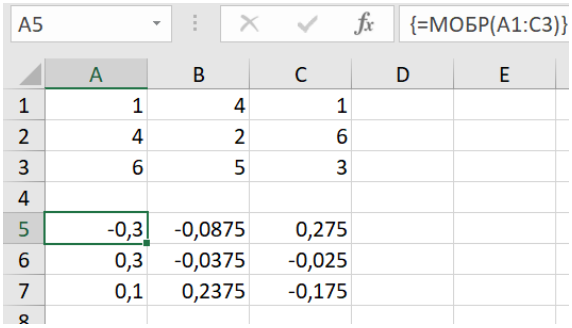


Рисунок 7.2 – Результат у MS Excel

Оскільки результат виконання збігається з результатом в MS Excel (рисунок 7.2), то даний метод працює вірно.

б) Метод розбиття на клітки.

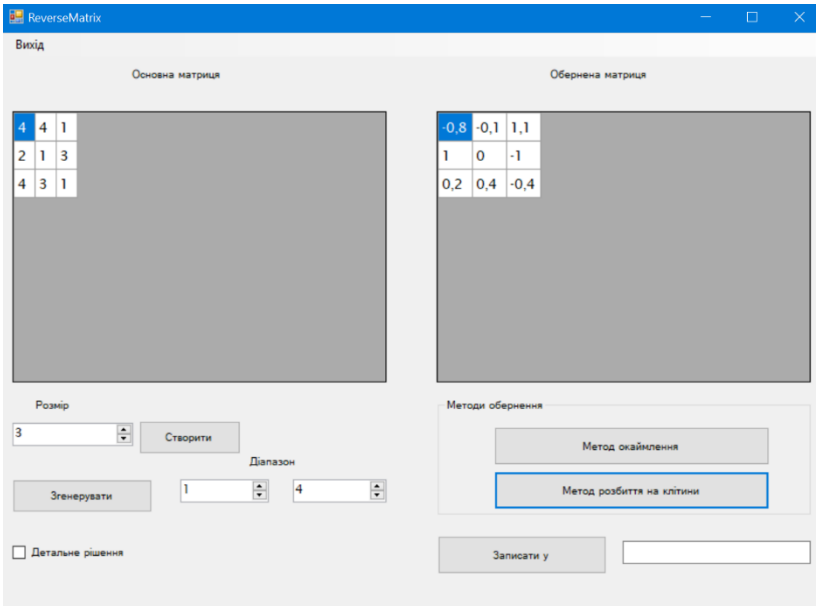


Рисунок 7.3 – Результат методу розбиття на клітки

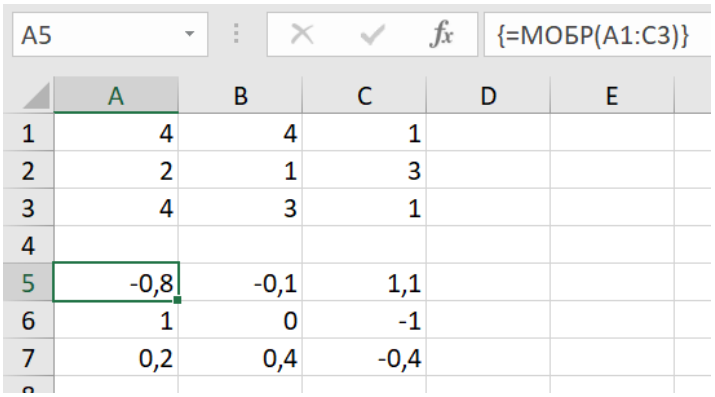
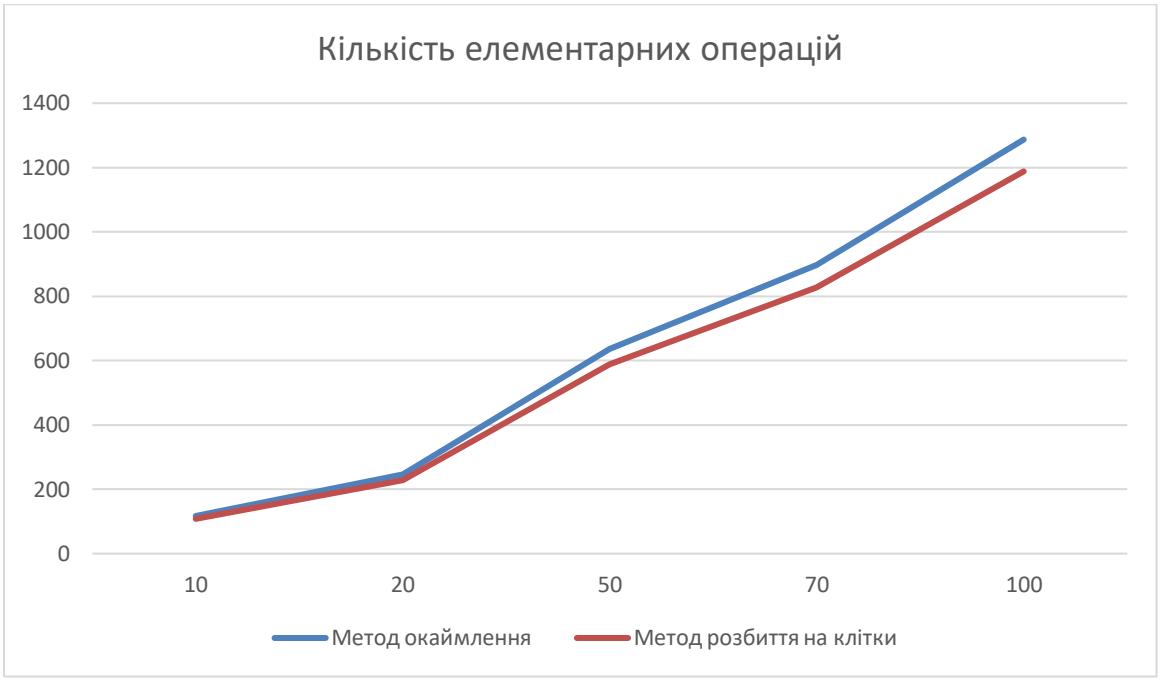


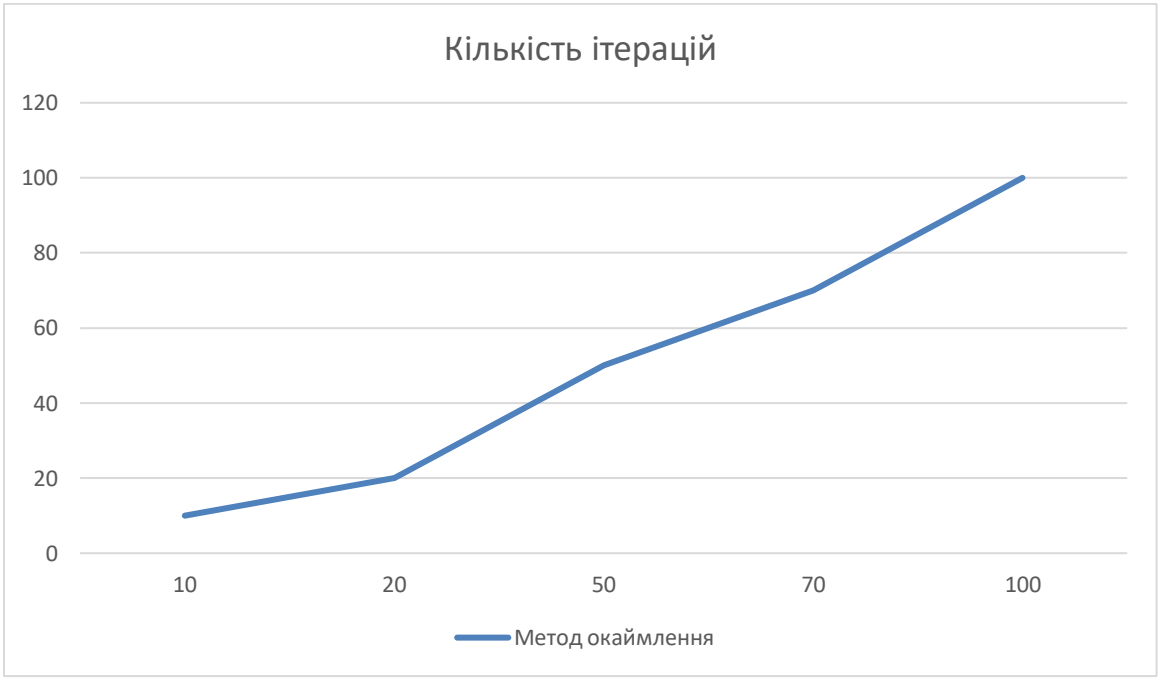
Рисунок 7.4 – Результат MS Excel

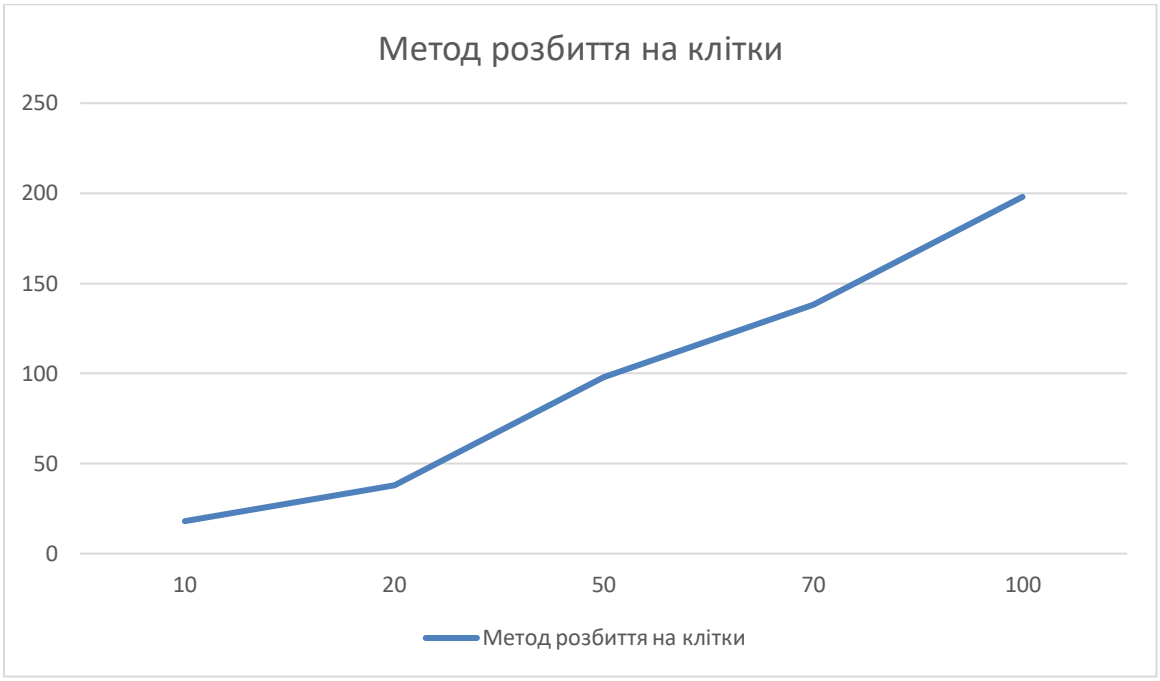
Оскільки результат виконання збігається з результатом в MS Excel (рисунок 7.4), то даний метод працює вірно.

Для перевірки ефективності програми буде використаний консольний варіант програми, з доданими лічильниками для підрахунку операцій.

Розмірність матриці	Параметри тестування	Метод	
		Окаймлення	Розбиття на клітки
10	Кількість ітерацій	10	-
	Кількість елементарних операцій	117	108
	Кількість рекурсій	-	18
20	Кількість ітерацій	20	-
	Кількість елементарних операцій	247	228
	Кількість рекурсій	-	38
50	Кількість ітерацій	50	-
	Кількість елементарних операцій	637	588
	Кількість рекурсій	-	98
70	Кількість ітерацій	70	-
	Кількість елементарних операцій	897	828
	Кількість рекурсій	-	138
100	Кількість ітерацій	100	-
	Кількість елементарних операцій	1287	1188
	Кількість рекурсій	-	198


Кількість елементарних операцій
1400
1200
1000
800
600
400
200
0
10
20
50
70
100
Метод окаймлення
Метод розбиття на клітки


Кількість ітерацій
120
100
80
60
40
20
0
10
20
50
70
100
Метод окаймлення



За результатами тестування можна зробити такі висновки:

а) Обидва методи дозволяють обертати квадратні невироджені матриці будь-якої розмірності.

б) Складність обох методів – лінійна, O(n), де n розмірність матриці.

в) З розглянутих методів найоптимальнішим для практичного використання є метод розбиття на клітки, оскільки він виконується найшвидше та легший для розуміння.

ВИСНОВОК

У даній роботі були досліджені два методи обернення матриці: метод окаймлення та розбиття на клітки. Було створено програмне забезпечення, яке обертає задану матрицю даними методами та відображає детальне рішення задачі. Для методу окаймлення були відображені результати по кожній ітерації, а для методу розбиття на клітки, відображені клітки та кількість рекурсій. Для візуалізації операцій та результатів був спроектований та створений графічний інтерфейс програми. Програмне забезпечення було створено за допомогою об’єктного-орієнтованого програмування. Були поставлені численні тестування програмного забезпечення. Результати програми виявилися правильними, що стверджує на її дієвість. Робота була зроблена за допомогою мови програмування C++.

ПЕРЕЛІК ПОСИЛАНЬ

1. Матриця (математика) URL: http://surl.li/cdirf
2. Метод розбиття на клітки URL: https://www.mathros.net.ua/znahodzhennja-obernenoi-matryci-vykorystovujuchy-metod-rozbyttja-na-klityny.html
3. Метод окаймлення URL: https://studfile.net/preview/4348938/page:9/

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії

Затвердив
Керівник Головченко Максим Миколайович
«___»_____2022 р.

Виконавець:
Студент Медвідь Олександр Русланович
«___» _____2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання курсової роботи
на тему: «Обернення матриці методами
окаймлення та розбиття на клітки»
з дисципліни:
«Основи програмування»

Київ 2022

1. *Мета*: Метою курсової роботи є розробка ефективного програмного забезпечення для розв’язання задачі обернення матриці методами окаймлення та розбиття на клітки.
2. *Дата початку роботи*: «02» травня 2022р
3. *Дата закінчення роботи*: «12» червня 2022 р.
4. *Вимоги до програмного забезпечення*.
   1) Функціональні вимоги:
      – Можливість ручного задання значень та розміру матриці
      – Можливість автоматичної генерації матриці, за вказаним діапазоном значень
      – Можливість вибору методу обернення матриці
      – Можливість обернення матриці обраним методом
      – Можливість графічного відображення матриці для ілюстрації розв’язку задачі
      – Можливість збереження результатів розв’язання задачі у файл
      – Можливість перегляду детального рішення задачі
   2) Нефункціональні вимоги:
      – Можливість запускати програмне забезпечення на операційній системі сімейства Windows
      – Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:
      ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.
      ГОСТ 19.106 - 78 - Вимоги до програмної документації.
      ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

5. *Стадії та етапи розробки*:
   1) Об'єктно-орієнтований аналіз предметної області задачі (до__.__.202_ р.)
   2) Об'єктно-орієнтоване проектування архітектури програмної системи (до __.__.202_р.)
   3) Розробка програмного забезпечення (до __.__.202_р.)
   4) Тестування розробленої програми (до __.__.202_р.)
   5) Розробка пояснювальної записки (до __.__.202_ р.).
   6) Захист курсової роботи (до __.__.202_ р.).
6. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду "Обернення матриці методами окаймлення та розбиття на клітки"

(Найменування програми(документа))

Електронний носій

(Вид носія даних)

94 арк, 211 Кб

(Обсяг програми (документа), арк., Кб)

студента групи ІП-1 4 курсу
Медвідя О.Р.

Файл RevMatForm.h

```
#pragma once
#include "Matrix.h"
#include "CellDivisionSolution.h"
#include "EmborderingSolution.h"

namespace ReverseMatrix
{

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для RevMatForm
	/// </summary>
	public ref class RevMatForm : public System::Windows::Forms::Form
	{
	public:
		RevMatForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
```

```
        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        ~RevMatForm()
        {
                if (components)
                {
                        delete components;
                }
        }

protected:


private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::DataGridView^ ReverseMatrixGridView;

private: System::Windows::Forms::DataGridView^ MainMatrixGridView;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::NumericUpDown^ MatrixSizeUpDown;
private: System::Windows::Forms::Button^ CreateMatrixButton;
private: System::Windows::Forms::Button^ ArrGenAutoButton;




private: System::Windows::Forms::GroupBox^ groupBox1;
private: System::Windows::Forms::Button^ EmborderingButton;
private: System::Windows::Forms::Button^ CellDivisionButton;
private: System::Windows::Forms::NumericUpDown^ MinNumberUpDown;
private: System::Windows::Forms::NumericUpDown^ MaxNumberUpDown;
```

```
        private: System::Windows::Forms::Label^ label3;
        private: System::Windows::Forms::Label^ label4;
        private: System::Windows::Forms::TextBox^ FileNameTextBox;
        private: System::Windows::Forms::Button^ WriteFileButton;
        private: System::Windows::Forms::MenuStrip^ menuStrip1;
        private: System::Windows::Forms::ToolStripMenuItem^
вихідToolStripMenuItem;
        private: System::Windows::Forms::CheckBox^ SolutionCheck;


        protected:


        private:
            /// <summary>
            /// Обязательная переменная конструктора.
            /// </summary>
            System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Требуемый метод для поддержки конструктора — не изменяйте
            /// содержимое этого метода с помощью редактора кода.
            /// </summary>
            void InitializeComponent(void)
            {
                this->label2 = (gcnew System::Windows::Forms::Label());
                this->ReverseMatrixGridView = (gcnew
System::Windows::Forms::DataGridView());
                this->MainMatrixGridView = (gcnew
System::Windows::Forms::DataGridView());
```

```
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->MatrixSizeUpDown = (gcnew
System::Windows::Forms::NumericUpDown());
            this->CreateMatrixButton = (gcnew
System::Windows::Forms::Button());
            this->ArrGenAutoButton = (gcnew
System::Windows::Forms::Button());
            this->groupBox1 = (gcnew
System::Windows::Forms::GroupBox());
            this->CellDivisionButton = (gcnew
System::Windows::Forms::Button());
            this->EmborderingButton = (gcnew
System::Windows::Forms::Button());
            this->MinNumberUpDown = (gcnew
System::Windows::Forms::NumericUpDown());
            this->MaxNumberUpDown = (gcnew
System::Windows::Forms::NumericUpDown());
            this->label3 = (gcnew System::Windows::Forms::Label());
            this->label4 = (gcnew System::Windows::Forms::Label());
            this->FileNameTextBox = (gcnew
System::Windows::Forms::TextBox());
            this->WriteFileButton = (gcnew
System::Windows::Forms::Button());
            this->menuStrip1 = (gcnew
System::Windows::Forms::MenuStrip());
            this->вихідToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->SolutionCheck = (gcnew
System::Windows::Forms::CheckBox());

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
```

```
>ReverseMatrixGridView))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MainMatrixGridView))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixSizeUpDown))->BeginInit();
                this->groupBox1->SuspendLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MinNumberUpDown))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MaxNumberUpDown))->BeginInit();
                this->menuStrip1->SuspendLayout();
                this->SuspendLayout();
                //
                // label2
                //
                this->label2->AutoSize = true;
                this->label2->Location = System::Drawing::Point(544, 33);
                this->label2->Name = L"label2";
                this->label2->Size = System::Drawing::Size(171, 23);
                this->label2->TabIndex = 3;
                this->label2->Text = L"Обернена матриця";
                //
                // ReverseMatrixGridView
                //
                this->ReverseMatrixGridView->AllowUserToAddRows = false;
                this->ReverseMatrixGridView->AllowUserToDeleteRows = false;
                this->ReverseMatrixGridView->AllowUserToResizeColumns =
```

```
false;
                this->ReverseMatrixGridView->AllowUserToResizeRows =
false;
                this->ReverseMatrixGridView->ColumnHeadersHeightSizeMode
=
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
                this->ReverseMatrixGridView->ColumnHeadersVisible = false;
                this->ReverseMatrixGridView->Location =
System::Drawing::Point(434, 78);
                this->ReverseMatrixGridView->Name =
L"ReverseMatrixGridView";
                this->ReverseMatrixGridView->ReadOnly = true;
                this->ReverseMatrixGridView->RowHeadersVisible = false;
                this->ReverseMatrixGridView->RowHeadersWidth = 62;
                this->ReverseMatrixGridView->RowTemplate->Height = 28;
                this->ReverseMatrixGridView->Size =
System::Drawing::Size(372, 269);
                this->ReverseMatrixGridView->TabIndex = 1;
                //
                // MainMatrixGridView
                //
                this->MainMatrixGridView->AllowUserToAddRows = false;
                this->MainMatrixGridView->AllowUserToDeleteRows = false;
                this->MainMatrixGridView->AllowUserToResizeColumns =
false;
                this->MainMatrixGridView->AllowUserToResizeRows = false;
                this->MainMatrixGridView->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
                this->MainMatrixGridView->ColumnHeadersVisible = false;
```

```
this->MainMatrixGridView->Location =
System::Drawing::Point(12, 78);
this->MainMatrixGridView->Name = L"MainMatrixGridView";
this->MainMatrixGridView->RowHeadersVisible = false;
this->MainMatrixGridView->RowHeadersWidth = 62;
this->MainMatrixGridView->RowTemplate->Height = 28;
this->MainMatrixGridView->Size = System::Drawing::Size(372,
269);
this->MainMatrixGridView->TabIndex = 0;
//
// label1
//
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(128, 33);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(160, 23);
this->label1->TabIndex = 2;
this->label1->Text = L"Основна матриця";
//
// MatrixSizeUpDown
//
this->MatrixSizeUpDown->Location =
System::Drawing::Point(12, 387);
this->MatrixSizeUpDown->Maximum = System::Decimal(gcnew
cli::array< System::Int32 >(4) { 6, 0, 0, 0 });
this->MatrixSizeUpDown->Minimum = System::Decimal(gcnew
cli::array< System::Int32 >(4) { 1, 0, 0, 0 });
this->MatrixSizeUpDown->Name = L"MatrixSizeUpDown";
this->MatrixSizeUpDown->Size = System::Drawing::Size(120,
31);
this->MatrixSizeUpDown->TabIndex = 4;
```

```
            this->MatrixSizeUpDown->Value = System::Decimal(gcnew
cli::array< System::Int32 >(4) { 1, 0, 0, 0 });
            //
            // CreateMatrixButton
            //
            this->CreateMatrixButton->Location =
System::Drawing::Point(138, 385);
            this->CreateMatrixButton->Name = L"CreateMatrixButton";
            this->CreateMatrixButton->Size = System::Drawing::Size(101,
33);
            this->CreateMatrixButton->TabIndex = 5;
            this->CreateMatrixButton->Text = L"Створити";
            this->CreateMatrixButton->UseVisualStyleBackColor = true;
            this->CreateMatrixButton->Click += gcnew
System::EventHandler(this, &RevMatForm::CreateMatrixButton_Click);
            //
            // ArrGenAutoButton
            //
            this->ArrGenAutoButton->Location =
System::Drawing::Point(12, 443);
            this->ArrGenAutoButton->Name = L"ArrGenAutoButton";
            this->ArrGenAutoButton->Size = System::Drawing::Size(139,
33);
            this->ArrGenAutoButton->TabIndex = 6;
            this->ArrGenAutoButton->Text = L"Згенерувати";
            this->ArrGenAutoButton->UseVisualStyleBackColor = true;
            this->ArrGenAutoButton->Click += gcnew
System::EventHandler(this, &RevMatForm::ArrGenAutoButton_Click);
            //
            // groupBox1
            //
```

```
            this->groupBox1->Controls->Add(this->CellDivisionButton);
            this->groupBox1->Controls->Add(this->EmborderingButton);
            this->groupBox1->Location = System::Drawing::Point(435, 361);
            this->groupBox1->Name = L"groupBox1";
            this->groupBox1->Size = System::Drawing::Size(371, 118);
            this->groupBox1->TabIndex = 9;
            this->groupBox1->TabStop = false;
            this->groupBox1->Text = L"Методи обернення";
            //
            // CellDivisionButton
            //
            this->CellDivisionButton->Location =
System::Drawing::Point(56, 74);
            this->CellDivisionButton->Name = L"CellDivisionButton";
            this->CellDivisionButton->Size = System::Drawing::Size(274,
38);
            this->CellDivisionButton->TabIndex = 1;
            this->CellDivisionButton->Text = L"Метод розбиття на
клітини";
            this->CellDivisionButton->UseVisualStyleBackColor = true;
            this->CellDivisionButton->Click += gcnew
System::EventHandler(this, &RevMatForm::CellDivisionButton_Click);
            //
            // EmborderingButton
            //
            this->EmborderingButton->Location =
System::Drawing::Point(56, 30);
            this->EmborderingButton->Name = L"EmborderingButton";
            this->EmborderingButton->Size = System::Drawing::Size(274,
38);
            this->EmborderingButton->TabIndex = 0;
```

```
            this->EmborderingButton->Text = L"Метод окаймлення";
            this->EmborderingButton->UseVisualStyleBackColor = true;
            this->EmborderingButton->Click += gcnew
System::EventHandler(this, &RevMatForm::EmborderingButton_Click);
            //
            // MinNumberUpDown
            //
            this->MinNumberUpDown->Location =
System::Drawing::Point(179, 443);
            this->MinNumberUpDown->Minimum = System::Decimal(gcnew
cli::array< System::Int32 >(4) { 1, 0, 0, 0 });
            this->MinNumberUpDown->Name = L"MinNumberUpDown";
            this->MinNumberUpDown->Size = System::Drawing::Size(88,
31);
            this->MinNumberUpDown->TabIndex = 10;
            this->MinNumberUpDown->Value = System::Decimal(gcnew
cli::array< System::Int32 >(4) { 1, 0, 0, 0 });
            //
            // MaxNumberUpDown
            //
            this->MaxNumberUpDown->Location =
System::Drawing::Point(291, 443);
            this->MaxNumberUpDown->Name = L"MaxNumberUpDown";
            this->MaxNumberUpDown->Size = System::Drawing::Size(93,
31);
            this->MaxNumberUpDown->TabIndex = 11;
            //
            // label3
            //
            this->label3->AutoSize = true;
            this->label3->Location = System::Drawing::Point(245, 417);
```

```
            this->label3->Name = L"label3";
            this->label3->Size = System::Drawing::Size(86, 23);
            this->label3->TabIndex = 12;
            this->label3->Text = L"Діапазон";
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->Location = System::Drawing::Point(32, 361);
            this->label4->Name = L"label4";
            this->label4->Size = System::Drawing::Size(66, 23);
            this->label4->TabIndex = 13;
            this->label4->Text = L"Розмір";
            //
            // FileNameTextBox
            //
            this->FileNameTextBox->Location =
System::Drawing::Point(619, 504);
            this->FileNameTextBox->Name = L"FileNameTextBox";
            this->FileNameTextBox->Size = System::Drawing::Size(187, 31);
            this->FileNameTextBox->TabIndex = 14;
            //
            // WriteFileButton
            //
            this->WriteFileButton->Location = System::Drawing::Point(435,
499);
            this->WriteFileButton->Name = L"WriteFileButton";
            this->WriteFileButton->Size = System::Drawing::Size(168, 38);
            this->WriteFileButton->TabIndex = 15;
            this->WriteFileButton->Text = L"Записати у";
            this->WriteFileButton->UseVisualStyleBackColor = true;
```

```
            this->WriteFileButton->Click += gcnew
System::EventHandler(this, &RevMatForm::WriteFileButton_Click);
            //
            // menuStrip1
            //
            this->menuStrip1->GripMargin =
System::Windows::Forms::Padding(2, 2, 0, 2);
            this->menuStrip1->ImageScalingSize =
System::Drawing::Size(24, 24);
            this->menuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) { this->вихідToolStripMenuItem });
            this->menuStrip1->Location = System::Drawing::Point(0, 0);
            this->menuStrip1->Name = L"menuStrip1";
            this->menuStrip1->Size = System::Drawing::Size(818, 33);
            this->menuStrip1->TabIndex = 16;
            this->menuStrip1->Text = L"menuStrip1";
            //
            // вихідToolStripMenuItem
            //
            this->вихідToolStripMenuItem->Name =
L"вихідToolStripMenuItem";
            this->вихідToolStripMenuItem->Size =
System::Drawing::Size(70, 29);
            this->вихідToolStripMenuItem->Text = L"Вихід";
            this->вихідToolStripMenuItem->Click += gcnew
System::EventHandler(this, &RevMatForm::вихідToolStripMenuItem_Click);
            //
            // SolutionCheck
            //
            this->SolutionCheck->AutoSize = true;
            this->SolutionCheck->Location = System::Drawing::Point(12,
```

```
506);
                this->SolutionCheck->Name = L"SolutionCheck";
                this->SolutionCheck->Size = System::Drawing::Size(193, 27);
                this->SolutionCheck->TabIndex = 17;
                this->SolutionCheck->Text = L"Детальне рішення";
                this->SolutionCheck->UseVisualStyleBackColor = true;
                //
                // RevMatForm
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(12, 23);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(818, 571);
                this->Controls->Add(this->SolutionCheck);
                this->Controls->Add(this->WriteFileButton);
                this->Controls->Add(this->FileNameTextBox);
                this->Controls->Add(this->label4);
                this->Controls->Add(this->label3);
                this->Controls->Add(this->MaxNumberUpDown);
                this->Controls->Add(this->MinNumberUpDown);
                this->Controls->Add(this->groupBox1);
                this->Controls->Add(this->label2);
                this->Controls->Add(this->ArrGenAutoButton);
                this->Controls->Add(this->ReverseMatrixGridView);
                this->Controls->Add(this->CreateMatrixButton);
                this->Controls->Add(this->MainMatrixGridView);
                this->Controls->Add(this->MatrixSizeUpDown);
                this->Controls->Add(this->label1);
                this->Controls->Add(this->menuStrip1);
                this->Font = (gcnew System::Drawing::Font(L"Lucida Sans", 10,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
```

```
                    static_cast<System::Byte>(0)));
                this->MainMenuStrip = this->menuStrip1;
                this->Margin = System::Windows::Forms::Padding(5, 4, 5, 4);
                this->Name = L"RevMatForm";
                this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
                this->Text = L"ReverseMatrix";

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>ReverseMatrixGridView))->EndInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MainMatrixGridView))->EndInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixSizeUpDown))->EndInit();
                this->groupBox1->ResumeLayout(false);

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MinNumberUpDown))->EndInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MaxNumberUpDown))->EndInit();
                this->menuStrip1->ResumeLayout(false);
                this->menuStrip1->PerformLayout();
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
```

```
private: System::Void CreateMatrixButton_Click(System::Object^ sender,
System::EventArgs^ e);
private: System::Void ArrGenAutoButton_Click(System::Object^ sender,
System::EventArgs^ e);
private: System::Void EmborderingButton_Click(System::Object^ sender,
System::EventArgs^ e);
private: System::Void CellDivisionButton_Click(System::Object^ sender,
System::EventArgs^ e);
private: void ShowMatrix(int SizeMatrix, Matrix MainMatrix, int type);
private: Matrix GetMatrix(int type);
private: System::Void WriteFileButton_Click(System::Object^ sender,
System::EventArgs^ e);
private: System::Void вихідToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e);
private: bool BannedSymbols();
};
}
```

Файл RevMatForm.cpp

```
#include "RevMatForm.h"
#include "Matrix.h"
#include "RevMat.h"
```

```
#include <string>
#include <msclr/marshal_cppstd.h>

using namespace std;
using namespace msclr::interop;
using namespace ReverseMatrix;

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
   Application::EnableVisualStyles();
   Application::SetCompatibleTextRenderingDefault(false);
   Application::Run(gcnew RevMatForm);
   return 0;
}

//Створення матриці
System::Void
ReverseMatrix::RevMatForm::CreateMatrixButton_Click(System::Object^ sender,
System::EventArgs^ e)
{
   int SizeMatrix;
   SizeMatrix = Convert::ToInt32(MatrixSizeUpDown->Value);
   Matrix MainMatrix(SizeMatrix, SizeMatrix);
   MainMatrix.ArrGenAuto(0, 0);

   MainMatrixGridView->RowCount = SizeMatrix;
   MainMatrixGridView->ColumnCount = SizeMatrix;

   ShowMatrix(SizeMatrix, MainMatrix, 1); //Виводим матрицю
   //Ячейки
   MainMatrixGridView-
```

```
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);


    return System::Void();
}

//Генерація матриці випадковим чином
System::Void
ReverseMatrix::RevMatForm::ArrGenAutoButton_Click(System::Object^ sender,
System::EventArgs^ e)
{
    if (MainMatrixGridView->RowCount == 0)
    {
        MessageBox::Show("Матриця не створена");
        return;
    }

    int MaxNumber, MinNumber;
    MaxNumber = Convert::ToInt32(MaxNumberUpDown->Value);
    MinNumber = Convert::ToInt32(MinNumberUpDown->Value);

    if (MinNumber == MaxNumber || MinNumber > MaxNumber)
    {
        MessageBox::Show("Заданий діапазон не може згенерувати невироджену
матрицю");
        return;
    }

    Matrix MainMatrix(MainMatrixGridView->RowCount, MainMatrixGridView-
>ColumnCount);
```

```
    MainMatrix.ArrGenAuto(MinNumber, MaxNumber);

    ShowMatrix(MainMatrix.GetRow(), MainMatrix, 1); //Виводим матрицю
    //Ячейки
    MainMatrixGridView-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
    MainMatrixGridView->AutoResizeColumns();//Стовбці
    return System::Void();
}

//Обернення матриці методом окаймлення
System::Void
ReverseMatrix::RevMatForm::EmborderingButton_Click(System::Object^ sender,
System::EventArgs^ e)
{
    if (BannedSymbols())
    {
        MessageBox::Show("Задана матриця містить некоректний символ");
        return;
    }

    Matrix  MainMatrix = GetMatrix(1);
    if (UnAcceptable(MainMatrix, 1))
    {
        MessageBox::Show("Задана матриця не підходить для даного методу");
        return;
    }

    if (SolutionCheck->Checked)
    {
```

```
        EmborderingSolution^ Form = gcnew EmborderingSolution(MainMatrix);
        Form->Show();
    }

    Matrix ReverseMatrix = Embordering(MainMatrix);
    ReverseMatrixGridView->RowCount = ReverseMatrix.GetRow();
    ReverseMatrixGridView->ColumnCount = ReverseMatrix.GetRow();

    ShowMatrix(ReverseMatrix.GetRow(), ReverseMatrix, 2);
    //Ячейки
    MainMatrixGridView-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
    MainMatrixGridView->AutoResizeColumns();//Стовбці

    return System::Void();
}

//Знаходженн матриці методом робиття на клітки
System::Void
ReverseMatrix::RevMatForm::CellDivisionButton_Click(System::Object^ sender,
System::EventArgs^ e)
{
    int count = 0;
    bool ZeroDiv = false;
    if (BannedSymbols())
    {
        MessageBox::Show("Задана матриця містить некоректний символ");
        return;
    }
```

```
    Matrix  MainMatrix = GetMatrix(1);

    if (UnAcceptable(MainMatrix, 2))
    {
        MessageBox::Show("Задана матриця не підходить для даного методу");
        return;
    }

    if (SolutionCheck->Checked)
    {
        if (MainMatrix.GetRow() == 1)
        {
            MessageBox::Show("Для матриці розмірності 1x1 детальне рішення
непотрібне");
        }
        else
        {
            CellDivisionSolution^ Form = gcnew CellDivisionSolution(MainMatrix);
            Form->Show();
        }

    }

    Matrix ReverseMatrix = CellDivision(MainMatrix, &count, &ZeroDiv);
    ReverseMatrixGridView->RowCount = ReverseMatrix.GetRow();
    ReverseMatrixGridView->ColumnCount = ReverseMatrix.GetRow();

    ShowMatrix(ReverseMatrix.GetRow(), ReverseMatrix, 2);
    //Ячейки
    ReverseMatrixGridView-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
```

```
zeToAllHeaders);
    ReverseMatrixGridView->AutoResizeColumns();//Стовбці

    return System::Void();
}

//Вивод матриці
void ReverseMatrix::RevMatForm::ShowMatrix(int SizeMatrix, Matrix MainMatrix,
int type)
{
    if (type == 1)
    {
        for (int i = 0; i < SizeMatrix; i++)
        {
            for (int j = 0; j < SizeMatrix; j++)
            {
                //Значення матриці
                MainMatrixGridView->Rows[i]->Cells[j]->Value =
MainMatrix.GetArr()[i][j];
            }
        }
    }
    else
    {
        for (int i = 0; i < SizeMatrix; i++)
        {
            for (int j = 0; j < SizeMatrix; j++)
            {
                //Значення матриці
                ReverseMatrixGridView->Rows[i]->Cells[j]->Value =
MainMatrix.GetArr()[i][j];
```

```
            }
        }
    }
}

//Зчитування матриці з DataGridView
Matrix ReverseMatrix::RevMatForm::GetMatrix(int type)
{
    Matrix SomeMatrix(MainMatrixGridView->RowCount, MainMatrixGridView-
>ColumnCount);

    if (type == 1)
    {
        for (int i = 0; i < MainMatrixGridView->RowCount; i++)
        {
            for (int j = 0; j < MainMatrixGridView->ColumnCount; j++)
            {
                SomeMatrix.GetArr()[i][j] = Convert::ToDouble(MainMatrixGridView-
>Rows[i]->Cells[j]->Value);
            }
        }
    }
    else
    {
        for (int i = 0; i < ReverseMatrixGridView->RowCount; i++)
        {
            for (int j = 0; j < ReverseMatrixGridView->ColumnCount; j++)
            {
                SomeMatrix.GetArr()[i][j] = Convert::ToDouble(ReverseMatrixGridView-
>Rows[i]->Cells[j]->Value);
            }
```

```
        }
    }

    return SomeMatrix;
}

//Запис оберненої матриці у файл
System::Void ReverseMatrix::RevMatForm::WriteFileButton_Click(System::Object^
sender, System::EventArgs^ e)
{
    Matrix ReverseMatrix = GetMatrix(2);
    string FileName = marshal_as<string>(FileNameTextBox->Text);

    int result = WriteFile(ReverseMatrix, FileName);
    if (result == 1)
    {
        MessageBox::Show("Файл не знайдений");
        return;
    }
    else
    {
        MessageBox::Show("Матриця успішно записана у файл ",
FileNameTextBox->Text);
        return;
    }
    return System::Void();
}


//Вихід
System::Void
```

```
ReverseMatrix::RevMatForm::вихідToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^ e)
{
   Application::Exit();
}

//Перевірка матриці на некоректні символи
bool ReverseMatrix::RevMatForm::BannedSymbols()
{
   string TrueSymbols = "0 1 2 3 4 5 6 7 8 9";
   string minus = "-";
   string comma = ",";
   string zero = "0";
   string Check;

   for (int i = 0; i < MainMatrixGridView->RowCount; i++)
   {
      for (int j = 0; j < MainMatrixGridView->ColumnCount; j++)
      {
         bool OneComma = false;
         Check = marshal_as<string>(Convert::ToString(MainMatrixGridView-
>Rows[i]->Cells[j]->Value));
         for (int k = 0; k < Check.length(); k++)
         {
            if (TrueSymbols.find(Check[k]) == string::npos)
            {
               //Якщо перший символ -
               if (k == 0 && minus.find(Check[k]) != string::npos)
               {
                  //Якщо перший символ -, а другий ,
                  if (comma.find(Check[k+1]) != string::npos)
```

```
                {
                    return true;
                }
                continue;
            }
            //Якщо перший символ ,
            if (k == 0 && comma.find(Check[k]) != string::npos)
            {
                return true;
            }
            //Якщо це не перша ,
            if (k > 0 && comma.find(Check[k]) != string::npos && OneComma ==
false)
            {
                OneComma = true;
                continue;
            }
            return true;
        }
    }
  }
 }
```

Файл Matrix.h

```
#pragma once
class Matrix //Клас матриці
{
```

```
private:
        int Row; //Кількість рядків матриці
        int Column; //Кількість стовпців матриці
        double** Arr; //Представлення самої матриці

public:
        Matrix(int RowNumber, int ColumnNumber); //Конструктор класу
        ~Matrix(); //Деструктор класу
        Matrix(const Matrix& matrix); //Конструктор копіювання
        void ArrGenAuto(int RangeMin, int RangeMax); //Генерація матриці випаковим чином
        int GetRow(); //Гетер рядків
        int GetColumn(); //Гетер стовпців
        double** GetArr(); //Гетер матрциі
        int Det(); //Функція знаходження визначника матриці
        //Функція викреслення рядка та стовпця
        void GetMatr(Matrix CurrentMatrix, int IndRow, int IndCol);
        //Перевантаження оператора потокового виводу
        Matrix operator - (const Matrix& matrix); //Перевантаження оператора віднімання
        Matrix operator * (const Matrix& matrix); //Перевантаження оператора для множення двох матриць
        Matrix operator * (double number); //Перевантаження оператора для множення матриці на число
        void operator = (const Matrix& OtherMatrix); //Перевантаження оператора присвоєння
};
```

Файл Matrix.cpp

```
#include <iostream>
#include "Matrix.h"
#include "RevMat.h"
```

```
using namespace std;

//Функції класу Matrix

Matrix::Matrix(int RowNumber, int ColumnNumber) //Конструктор класу
{
	Row = RowNumber;
	Column = ColumnNumber;
	Arr = new double* [Row];

	for (int i = 0; i < Row; i++)
	{
		Arr[i] = new double[Column];
	}
}

Matrix::~Matrix() //Деструктор класу
{
	for (int i = 0; i < Row; i++)
	{
		delete[] Arr[i];
	}
	delete[] Arr;
}

Matrix::Matrix(const Matrix& matrix)  //Конструктор копіювання
{
	this->Row = matrix.Row;
	this->Column = matrix.Column;
	this->Arr = new double* [matrix.Row];
	for (int i = 0; i < matrix.Row; i++)
```

```
	{
		this->Arr[i] = new double[Column];
	}
	for (int i = 0; i < matrix.Row; i++)
	{
		for (int j = 0; j < matrix.Column; j++)
		{
			this->Arr[i][j] = matrix.Arr[i][j];
		}
	}
}

void Matrix::ArrGenAuto(int RangeMin, int RangeMax) //Генерація матриці випадковим чином
{
	srand(time(NULL));
	for (int k = 0; ; k++)
	{
		for (int i = 0; i < Row; i++)
		{
			for (int j = 0; j < Column; j++)
			{
				if (RangeMax == 0)
				{
					Arr[i][j] = 0;
				}
				else
				{
					Arr[i][j] = rand() % (RangeMax - RangeMin + 1) + RangeMin;
				}
```

```
                }
            }
            if (UnAcceptable(*this, 3) && RangeMax != 0)
            {
                continue;
            }
            else
            {
                break;
            }
        }
}

int Matrix::Det() //Функція знаходження визначника матриці
{
    int temp = 0;
    int k = 1;
    Matrix CurrentMatrix(*this);

    if (Row == 1)
    {
        temp = Arr[0][0];
    }
    else if (Row == 2)
    {
        temp = Arr[0][0] * Arr[1][1] - Arr[1][0] * Arr[0][1];
    }
    else
    {
        for (int i = 0; i < Row; i++)
        {
```

```
                int m = Row - 1;
                Matrix TempMatrix(m, m);
                TempMatrix.GetMatr(CurrentMatrix, 0, i);
                temp = temp + k * Arr[0][i] * TempMatrix.Det();
                k = -k;
            }
        }
        return temp;
}
//Функція викреслення рядка та стовпця
void Matrix::GetMatr(Matrix CurrentMatrix, int IndRow, int IndCol)
{
        int ki = 0;
        for (int i = 0; i < CurrentMatrix.Row; i++)
        {
            if (i != IndRow)
            {
                for (int j = 0, kj = 0; j < CurrentMatrix.Row; j++)
                {
                    if (j != IndCol)
                    {
                        Arr[ki][kj] = CurrentMatrix.Arr[i][j];
                        kj++;
                    }
                }
                ki++;
            }
        }
}

int Matrix::GetRow() //Гетер рядків матриці
```

```
{
	return Row;
}

int Matrix::GetColumn() //Гетер стовпців матриці
{
	return Column;
}

double** Matrix::GetArr() //Гетер матрциі
{
	return Arr;
}

Matrix Matrix::operator - (const Matrix& matrix) //Перевантаження оператора віднімання
{
	Matrix NewMatrix(matrix.Row, matrix.Column);
	for (int i = 0; i < Row; i++)
	{
		for (int j = 0; j < Column; j++)
		{
			NewMatrix.Arr[i][j] = this->Arr[i][j] - matrix.Arr[i][j];
		}
	}
	return NewMatrix;
}

Matrix Matrix::operator * (const Matrix& matrix) //Перевантаження оператора для множення двох матриць
{
```

```
	Matrix NewMatrix(Row, matrix.Column);
	NewMatrix.ArrGenAuto(0, 0);
	for (int i = 0; i < Row; i++)
	{
		for (int j = 0; j < matrix.Column; j++)
		{
			for (int k = 0; k < matrix.Row; k++)
			{
				NewMatrix.Arr[i][j] += this->Arr[i][k] * matrix.Arr[k][j];
			}
		}
	}
	return NewMatrix;
}

Matrix Matrix::operator * (double number) //Перевантаження оператора для множення матриці на число
{
	Matrix NewMatrix(Row, Column);
	for (int i = 0; i < Row; i++)
	{
		for (int j = 0; j < Column; j++)
		{
			NewMatrix.Arr[i][j] = this->Arr[i][j] * number;
		}
	}
	return NewMatrix;
}

void Matrix::operator = (const Matrix& OtherMatrix) //Перевантаження оператора присвоєння
```

```
{
	int OldRow = this->Row;
	this->Row = OtherMatrix.Row;
	this->Column = OtherMatrix.Column;

	if (this->Arr != nullptr)
	{
		for (int i = 0; i < OldRow; i++)
		{
			delete[] this->Arr[i];
		}
		delete[] this->Arr;
	}

	this->Arr = new double* [OtherMatrix.Row];
	for (int i = 0; i < OtherMatrix.Row; i++)
	{
		this->Arr[i] = new double[Column];
	}
	for (int i = 0; i < OtherMatrix.Row; i++)
	{
		for (int j = 0; j < OtherMatrix.Column; j++)
		{
			this->Arr[i][j] = OtherMatrix.Arr[i][j];
		}
	}
}
```

Файл RevMat.h

```
#pragma once
#include <string>
```

```
bool DoubleZero(double number); //Перевірка на машинні нулі у матриці
bool UnAcceptable(Matrix MainMatrix, int type); //Перевірка матриці на визначники різних її частин
Matrix MachineZero(Matrix CurrentMatrix); //Заміна усіх машинних нулей на звичайні
Matrix Embordering(Matrix MainMatrix); //Метод окаймлення
Matrix CellDivisionInitialization(Matrix MainMatrix, int number); //Розбиття матриці на клітки
//Зібрання кліток в одну
Matrix CellDivisionBuild(Matrix MainMatrix, Matrix MatrixR11, Matrix MatrixR12, Matrix MatrixR21, Matrix MatrixR22);
//Знаходження матриці методом розбиття на клітки
Matrix CellDivision(Matrix MainMatrix, int* count, bool* ZeroDiv);
int WriteFile(Matrix ReverseMatrix, std::string FileName); //Функція запису у файл
```

Файл RevMat.cpp

```
#include <iostream>
#include "Matrix.h"
#include "RevMat.h"
#include <ctime>
#include <cmath>
#include <string>
#include <fstream>
using namespace std;

bool UnAcceptable(Matrix MainMatrix, int type) //Перевірка матриці на визначники різних її частин
{
	if (MainMatrix.GetRow() == 1 && MainMatrix.GetColumn() == 1 && MainMatrix.GetArr()[0][0] != 0)
	{
```

```
        return false;
}
if (MainMatrix.Det() == 0)
{
        return true;
}
if (type == 1 || type == 3)
{
        if (MainMatrix.GetArr()[0][0] == 0)
        {
                return true;
        }
        for (int k = 2; k < MainMatrix.GetRow(); k++)
        {
                Matrix Temp(k, k);
                for (int i = 0; i < k; i++)
                {
                        for (int j = 0; j < k; j++)
                        {
                                Temp.GetArr()[i][j] = MainMatrix.GetArr()[i][j];
                        }
                }
                if (Temp.Det() == 0)
                {
                        return true;
                }
        }
}
if (type == 2 || type == 3)
{
        int count = 0;
```

```
            bool ZeroDiv = false;
            CellDivision(MainMatrix, &count, &ZeroDiv);
            if (ZeroDiv)
            {
                return true;
            }
        }
}

Matrix MachineZero(Matrix CurrentMatrix) //Заміна усіх машинних нулей на звичайні
{
        for (int i = 0; i < CurrentMatrix.GetRow(); i++)
        {
            for (int j = 0; j < CurrentMatrix.GetColumn(); j++)
            {
                if (DoubleZero(CurrentMatrix.GetArr()[i][j]))
                {
                    CurrentMatrix.GetArr()[i][j] = 0;
                }
            }
        }
        return CurrentMatrix;
}

bool DoubleZero(double number) //Перевірка на машинні нулі у матриці
{
        string ZeroCheck = to_string(number);
        string ZeroMinus = "-0.000000";
        string ZeroPlus = "0.000000";
        if (ZeroCheck == ZeroMinus)
```

```
	{
		return true;
	}
	if (ZeroCheck == ZeroPlus)
	{
		return true;
	}
	else
	{
		return false;
	}
}

Matrix Embordering(Matrix MainMatrix) //Метод окаймлення
{
	int Disposal = MainMatrix.GetRow();

	Matrix OldA(1, 1);
	OldA.GetArr()[0][0] = 1 / MainMatrix.GetArr()[0][0];
	if (MainMatrix.GetRow() == 1)
	{
		return OldA;
	}

	for (int k = 1; k < Disposal; k++)
	{
		Matrix A(k + 1, k + 1);
		for (int i = 0; i < k + 1; i++)
		{
			for (int j = 0; j < k + 1; j++)
			{
```

```
                A.GetArr()[i][j] = MainMatrix.GetArr()[i][j];
        }
}

Matrix U(k, 1);
Matrix V(1, k);
double a = A.GetArr()[k][k];
for (int i = 0; i < k; i++)
{
        U.GetArr()[i][0] = A.GetArr()[i][k];
        V.GetArr()[0][i] = A.GetArr()[k][i];
}
double akk = a - (V * OldA * U).GetArr()[0][0];
Matrix r = OldA * U * (-1 / akk);
Matrix q = V * OldA * (-1 / akk);
Matrix B = OldA - (OldA * U) * q;


for (int i = 0; i < k; i++)
{
        for (int j = 0; j < k; j++)
        {
                A.GetArr()[i][j] = B.GetArr()[i][j];
        }
}
for (int j = 0; j < k; j++)
{
        A.GetArr()[j][k] = r.GetArr()[j][0];
        A.GetArr()[k][j] = q.GetArr()[0][j];
}
A.GetArr()[k][k] = (1 / akk);
```

```
            if (k == Disposal - 1)
            {
                return MachineZero(A);
            }
            OldA = A;
        }
}

Matrix CellDivisionInitialization(Matrix MainMatrix, int number) //Розбиття матриці на клітки
{
        int Disposal = MainMatrix.GetRow();

        if (Disposal % 2 == 0)
        {
            Matrix MatrixCell11(Disposal / 2, Disposal / 2);
            Matrix MatrixCell12(Disposal / 2, Disposal / 2);
            Matrix MatrixCell21(Disposal / 2, Disposal / 2);
            Matrix MatrixCell22(Disposal / 2, Disposal / 2);

            for (int i = 0; i < Disposal; i++)
            {
                for (int j = 0; j < Disposal; j++)
                {
                    if (i < Disposal / 2 && j < Disposal / 2)
                    {
                        MatrixCell11.GetArr()[i][j] = MainMatrix.GetArr()[i][j];
                    }
                    else if (i < Disposal / 2 && j >= Disposal / 2)
                    {
```

```
                        MatrixCell12.GetArr()[i][j - Disposal / 2] =
MainMatrix.GetArr()[i][j];
                    }
                    else if (i >= Disposal / 2 && j < Disposal / 2)
                    {
                        MatrixCell21.GetArr()[i - Disposal / 2][j] =
MainMatrix.GetArr()[i][j];
                    }
                    else if (i >= Disposal / 2 && j >= Disposal / 2)
                    {
                        MatrixCell22.GetArr()[i - Disposal / 2][j - Disposal /
2] = MainMatrix.GetArr()[i][j];
                    }
                }
            }
            if (number == 1)
            {
                return MatrixCell11;
            }
            else if (number == 2)
            {
                return MatrixCell12;
            }
            else if (number == 3)
            {
                return MatrixCell21;
            }
            else
            {
                return MatrixCell22;
            }
```

```
        }
        else
        {
                Matrix MatrixCell11(Disposal / 2 + 1, Disposal / 2 + 1);
                Matrix MatrixCell12(Disposal / 2 + 1, Disposal / 2);
                Matrix MatrixCell21(Disposal / 2, Disposal / 2 + 1);
                Matrix MatrixCell22(Disposal / 2, Disposal / 2);

                for (int i = 0; i < Disposal; i++)
                {
                        for (int j = 0; j < Disposal; j++)
                        {
                                if (i < Disposal / 2 + 1 && j < Disposal / 2 + 1)
                                {
                                        MatrixCell11.GetArr()[i][j] =
MainMatrix.GetArr()[i][j];
                                }
                                else if (i < Disposal / 2 + 1 && j >= Disposal / 2 + 1)
                                {
                                        MatrixCell12.GetArr()[i][j - (Disposal / 2 + 1)] =
MainMatrix.GetArr()[i][j];
                                }
                                else if (i >= Disposal / 2 + 1 && j < Disposal / 2 + 1)
                                {
                                        MatrixCell21.GetArr()[i - (Disposal / 2 + 1)][j] =
MainMatrix.GetArr()[i][j];
                                }
                                else if (i >= Disposal / 2 + 1 && j >= Disposal / 2 + 1)
                                {
                                        MatrixCell22.GetArr()[i - (Disposal / 2 + 1)][j -
(Disposal / 2 + 1)] = MainMatrix.GetArr()[i][j];
```

```
                        }
                }
        }
        if (number == 1)
        {
                return MatrixCell11;
        }
        else if (number == 2)
        {
                return MatrixCell12;
        }
        else if (number == 3)
        {
                return MatrixCell21;
        }
        else
        {
                return MatrixCell22;
        }
    }
}

//Зібрання кліток в одну
Matrix CellDivisionBuild(Matrix MainMatrix, Matrix MatrixR11, Matrix MatrixR12,
Matrix MatrixR21, Matrix MatrixR22)
{
    Matrix ReverseMatrix(MainMatrix.GetRow(), MainMatrix.GetColumn());
    int Disposal = ReverseMatrix.GetRow();
    if (Disposal % 2 == 0)
    {
        for (int i = 0; i < Disposal; i++)
```

```
            {
                for (int j = 0; j < Disposal; j++)
                {
                    if (i < Disposal / 2 && j < Disposal / 2)
                    {
                        ReverseMatrix.GetArr()[i][j] =
MatrixR11.GetArr()[i][j];
                    }
                    else if (i < Disposal / 2 && j >= Disposal / 2)
                    {
                        ReverseMatrix.GetArr()[i][j] =
MatrixR12.GetArr()[i][j - Disposal / 2];
                    }
                    else if (i >= Disposal / 2 && j < Disposal / 2)
                    {
                        ReverseMatrix.GetArr()[i][j] = MatrixR21.GetArr()[i
- Disposal / 2][j];
                    }
                    else if (i >= Disposal / 2 && j >= Disposal / 2)
                    {
                        ReverseMatrix.GetArr()[i][j] = MatrixR22.GetArr()[i
- Disposal / 2][j - Disposal / 2];
                    }
                }
            }
        }
        else
        {
            for (int i = 0; i < Disposal; i++)
            {
                for (int j = 0; j < Disposal; j++)
```

```
                {
                        if (i < Disposal / 2 + 1 && j < Disposal / 2 + 1)
                        {
                                ReverseMatrix.GetArr()[i][j] =
MatrixR11.GetArr()[i][j];
                        }
                        else if (i < Disposal / 2 + 1 && j >= Disposal / 2 + 1)
                        {
                                ReverseMatrix.GetArr()[i][j] =
MatrixR12.GetArr()[i][j - (Disposal / 2 + 1)];
                        }
                        else if (i >= Disposal / 2 + 1 && j < Disposal / 2 + 1)
                        {
                                ReverseMatrix.GetArr()[i][j] = MatrixR21.GetArr()[i
- (Disposal / 2 + 1)][j];
                        }
                        else if (i >= Disposal / 2 + 1 && j >= Disposal / 2 + 1)
                        {
                                ReverseMatrix.GetArr()[i][j] = MatrixR22.GetArr()[i
- (Disposal / 2 + 1)][j - (Disposal / 2 + 1)];
                        }
                }
            }
        }
        return ReverseMatrix;
}


//Знаходження матриці методом розбиття на клітки
Matrix CellDivision(Matrix MainMatrix, int* count, bool* ZeroDiv)
{
```

```
        if (MainMatrix.Det() == 0)
        {
                (*ZeroDiv) = true;
        }
        if (MainMatrix.GetRow() == 1)
        {
                if (MainMatrix.GetArr()[0][0] == 0)
                {
                        (*ZeroDiv) = true;
                }
                MainMatrix.GetArr()[0][0] = 1 / MainMatrix.GetArr()[0][0];
                return MainMatrix;
        }

        Matrix MatrixCell11 = CellDivisionInitialization(MainMatrix, 1);
        Matrix MatrixCell12 = CellDivisionInitialization(MainMatrix, 2);
        Matrix MatrixCell21 = CellDivisionInitialization(MainMatrix, 3);
        Matrix MatrixCell22 = CellDivisionInitialization(MainMatrix, 4);
        (*count)++;
        Matrix MatrixCell22Inverse = CellDivision(MatrixCell22, count, ZeroDiv);

        (*count)++;
        Matrix MatrixR11 = CellDivision(MatrixCell11 - (MatrixCell12 *
(MatrixCell22Inverse * MatrixCell21)), count, ZeroDiv);
        Matrix MatrixR12 = ((MatrixR11 * (-1)) * MatrixCell12) *
MatrixCell22Inverse;
        Matrix MatrixR21 = (MatrixCell22Inverse * (-1)) * MatrixCell21 *
MatrixR11;
        Matrix MatrixR22 = MatrixCell22Inverse - (MatrixCell22Inverse *
MatrixCell21 * MatrixR12);
```

```
	Matrix ReverseMatrix = CellDivisionBuild(MainMatrix, MatrixR11, MatrixR12, MatrixR21, MatrixR22);
	return MachineZero(ReverseMatrix);
}

int WriteFile(Matrix ReverseMatrix, string FileName) //Функція запису у файл
{
	ofstream f1; //Переменная f1 отвечает за запись в файл
	f1.open(FileName); //Открываем файл для записи
	if (!f1.is_open()) //Проверка открытия файла
	{
		return 1;
	}
	else //Заполнения нового файла
	{
		f1 << ReverseMatrix.GetRow() << "\t";
		f1 << ReverseMatrix.GetColumn();

		for (int i = 0; i < ReverseMatrix.GetRow(); i++)
		{
			f1 << "\n";
			for (int j = 0; j < ReverseMatrix.GetColumn(); j++)
			{
				f1 << ReverseMatrix.GetArr()[i][j] << "\t";
			}
		}
	}
	f1.close();
	return 2;
}
```

Файл EmborderingSolution.h

```
#pragma once
#include "Matrix.h"

namespace ReverseMatrix {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для EmborderingSolution
	/// </summary>
	public ref class EmborderingSolution : public System::Windows::Forms::Form
	{
	public:
		EmborderingSolution(Matrix MainMatrix)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
			Esolution(MainMatrix, 0);
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
```

```
        /// </summary>
        ~EmborderingSolution()
        {
                if (components)
                {
                        delete components;
                }
        }

protected:

private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::DataGridView^ MatrixAView;

private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::DataGridView^ UView;
private: System::Windows::Forms::DataGridView^ VView;


private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::TextBox^ IterText;
private: System::Windows::Forms::TextBox^ AkkText;
private: System::Windows::Forms::TextBox^ AkText;


private: System::Windows::Forms::Label^ label5;
private: System::Windows::Forms::Label^ label6;
private: System::Windows::Forms::Label^ label7;
private: System::Windows::Forms::Label^ label8;
```

```
        private: System::Windows::Forms::DataGridView^ qView;

        private: System::Windows::Forms::DataGridView^ rView;

        private: System::Windows::Forms::Label^ label9;
        private: System::Windows::Forms::DataGridView^ BView;

        private: System::Windows::Forms::Label^ label10;
        private: System::Windows::Forms::DataGridView^ AkGrid;

        private: System::Windows::Forms::Button^ IterButton;
        private: System::Windows::Forms::Label^ label11;
        private: System::Windows::Forms::DataGridView^ MainMatrixE;
        private: System::Windows::Forms::MenuStrip^ menuStrip1;
        private: System::Windows::Forms::ToolStripMenuItem^
вихідToolStripMenuItem;



        protected:

        private:
            /// <summary>
            /// Обязательная переменная конструктора.
            /// </summary>
            System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Требуемый метод для поддержки конструктора — не изменяйте
            /// содержимое этого метода с помощью редактора кода.
            /// </summary>
```

```
        void InitializeComponent(void)
        {
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->MatrixAView = (gcnew
System::Windows::Forms::DataGridView());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->UView = (gcnew
System::Windows::Forms::DataGridView());
            this->VView = (gcnew
System::Windows::Forms::DataGridView());
            this->label3 = (gcnew System::Windows::Forms::Label());
            this->label4 = (gcnew System::Windows::Forms::Label());
            this->IterText = (gcnew System::Windows::Forms::TextBox());
            this->AkkText = (gcnew System::Windows::Forms::TextBox());
            this->AkText = (gcnew System::Windows::Forms::TextBox());
            this->label5 = (gcnew System::Windows::Forms::Label());
            this->label6 = (gcnew System::Windows::Forms::Label());
            this->label7 = (gcnew System::Windows::Forms::Label());
            this->label8 = (gcnew System::Windows::Forms::Label());
            this->qView = (gcnew
System::Windows::Forms::DataGridView());
            this->rView = (gcnew
System::Windows::Forms::DataGridView());
            this->label9 = (gcnew System::Windows::Forms::Label());
            this->BView = (gcnew
System::Windows::Forms::DataGridView());
            this->label10 = (gcnew System::Windows::Forms::Label());
            this->AkGrid = (gcnew
System::Windows::Forms::DataGridView());
            this->IterButton = (gcnew System::Windows::Forms::Button());
            this->label11 = (gcnew System::Windows::Forms::Label());
```

```
            this->MainMatrixE = (gcnew
System::Windows::Forms::DataGridView());
            this->menuStrip1 = (gcnew
System::Windows::Forms::MenuStrip());
            this->вихідToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixAView))->BeginInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>UView))->BeginInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>VView))->BeginInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->qView))-
>BeginInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->rView))-
>BeginInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>BView))->BeginInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>AkGrid))->BeginInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MainMatrixE))->BeginInit();
            this->menuStrip1->SuspendLayout();
```

```
this->SuspendLayout();
//
// label1
//
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(45, 194);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(34, 20);
this->label1->TabIndex = 111;
this->label1->Text = L"k = ";
//
// MatrixAView
//
this->MatrixAView->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
this->MatrixAView->ColumnHeadersVisible = false;
this->MatrixAView->Location = System::Drawing::Point(84,
262);
this->MatrixAView->Name = L"MatrixAView";
this->MatrixAView->ReadOnly = true;
this->MatrixAView->RowHeadersVisible = false;
this->MatrixAView->RowHeadersWidth = 62;
this->MatrixAView->RowTemplate->Height = 28;
this->MatrixAView->Size = System::Drawing::Size(339, 310);
this->MatrixAView->TabIndex = 112;
//
// label2
//
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(32, 392);
```

```
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(33, 20);
            this->label2->TabIndex = 113;
            this->label2->Text = L"A =";
            //
            // UView
            //
            this->UView->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->UView->ColumnHeadersVisible = false;
            this->UView->Location = System::Drawing::Point(1044, 365);
            this->UView->Name = L"UView";
            this->UView->ReadOnly = true;
            this->UView->RowHeadersVisible = false;
            this->UView->RowHeadersWidth = 62;
            this->UView->RowTemplate->Height = 28;
            this->UView->Size = System::Drawing::Size(117, 336);
            this->UView->TabIndex = 114;
            //
            // VView
            //
            this->VView->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->VView->ColumnHeadersVisible = false;
            this->VView->Location = System::Drawing::Point(468, 392);
            this->VView->Name = L"VView";
            this->VView->ReadOnly = true;
            this->VView->RowHeadersVisible = false;
            this->VView->RowHeadersWidth = 62;
```

```
this->VView->RowTemplate->Height = 28;
this->VView->Size = System::Drawing::Size(502, 111);
this->VView->TabIndex = 115;
//
// label3
//
this->label3->AutoSize = true;
this->label3->Location = System::Drawing::Point(987, 546);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(34, 20);
this->label3->TabIndex = 116;
this->label3->Text = L"U =";
//
// label4
//
this->label4->AutoSize = true;
this->label4->Location = System::Drawing::Point(429, 447);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(33, 20);
this->label4->TabIndex = 117;
this->label4->Text = L"V =";
//
// IterText
//
this->IterText->Location = System::Drawing::Point(84, 194);
this->IterText->Name = L"IterText";
this->IterText->ReadOnly = true;
this->IterText->Size = System::Drawing::Size(100, 26);
this->IterText->TabIndex = 118;
//
// AkkText
```

```
//
this->AkkText->Location = System::Drawing::Point(496, 546);
this->AkkText->Name = L"AkkText";
this->AkkText->ReadOnly = true;
this->AkkText->Size = System::Drawing::Size(100, 26);
this->AkkText->TabIndex = 119;
//
// AkText
//
this->AkText->Location = System::Drawing::Point(789, 546);
this->AkText->Name = L"AkText";
this->AkText->ReadOnly = true;
this->AkText->Size = System::Drawing::Size(100, 26);
this->AkText->TabIndex = 120;
//
// label5
//
this->label5->AutoSize = true;
this->label5->Location = System::Drawing::Point(441, 546);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(51, 20);
this->label5->TabIndex = 121;
this->label5->Text = L"akk = ";
//
// label6
//
this->label6->AutoSize = true;
this->label6->Location = System::Drawing::Point(637, 546);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(136, 20);
this->label6->TabIndex = 122;
```

```
            this->label6->Text = L"ak = akk - V A U =";
            //
            // label7
            //
            this->label7->AutoSize = true;
            this->label7->Location = System::Drawing::Point(26, 935);
            this->label7->Name = L"label7";
            this->label7->Size = System::Drawing::Size(113, 20);
            this->label7->TabIndex = 126;
            this->label7->Text = L"q = -1/ak V A =";
            //
            // label8
            //
            this->label8->AutoSize = true;
            this->label8->Location = System::Drawing::Point(168, 726);
            this->label8->Name = L"label8";
            this->label8->Size = System::Drawing::Size(110, 20);
            this->label8->TabIndex = 125;
            this->label8->Text = L"r = -1/ak A U =";
            //
            // qView
            //
            this->qView->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->qView->ColumnHeadersVisible = false;
            this->qView->Location = System::Drawing::Point(145, 894);
            this->qView->Name = L"qView";
            this->qView->RowHeadersVisible = false;
            this->qView->RowHeadersWidth = 62;
            this->qView->RowTemplate->Height = 28;
```

```
            this->qView->Size = System::Drawing::Size(552, 106);
            this->qView->TabIndex = 124;
            //
            // rView
            //
            this->rView->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->rView->ColumnHeadersVisible = false;
            this->rView->Location = System::Drawing::Point(296, 587);
            this->rView->Name = L"rView";
            this->rView->ReadOnly = true;
            this->rView->RowHeadersVisible = false;
            this->rView->RowHeadersWidth = 62;
            this->rView->RowTemplate->Height = 28;
            this->rView->Size = System::Drawing::Size(177, 301);
            this->rView->TabIndex = 123;
            //
            // label9
            //
            this->label9->AutoSize = true;
            this->label9->Location = System::Drawing::Point(589, 821);
            this->label9->Name = L"label9";
            this->label9->Size = System::Drawing::Size(128, 20);
            this->label9->TabIndex = 128;
            this->label9->Text = L"B = A - (A U) q = ";
            //
            // BView
            //
            this->BView->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
```

```
e;
                this->BView->ColumnHeadersVisible = false;
                this->BView->Location = System::Drawing::Point(723, 715);
                this->BView->Name = L"BView";
                this->BView->ReadOnly = true;
                this->BView->RowHeadersVisible = false;
                this->BView->RowHeadersWidth = 62;
                this->BView->RowTemplate->Height = 28;
                this->BView->Size = System::Drawing::Size(409, 285);
                this->BView->TabIndex = 127;
                //
                // label10
                //
                this->label10->AutoSize = true;
                this->label10->Location = System::Drawing::Point(1161, 859);
                this->label10->Name = L"label10";
                this->label10->Size = System::Drawing::Size(41, 20);
                this->label10->TabIndex = 130;
                this->label10->Text = L"Ak =";
                //
                // AkGrid
                //
                this->AkGrid->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
                this->AkGrid->ColumnHeadersVisible = false;
                this->AkGrid->Location = System::Drawing::Point(1208, 671);
                this->AkGrid->Name = L"AkGrid";
                this->AkGrid->ReadOnly = true;
                this->AkGrid->RowHeadersVisible = false;
                this->AkGrid->RowHeadersWidth = 62;
```

```
this->AkGrid->RowTemplate->Height = 28;
this->AkGrid->Size = System::Drawing::Size(432, 353);
this->AkGrid->TabIndex = 129;
//
// IterButton
//
this->IterButton->Location = System::Drawing::Point(1568, 512);
this->IterButton->Name = L"IterButton";
this->IterButton->Size = System::Drawing::Size(127, 54);
this->IterButton->TabIndex = 131;
this->IterButton->Text = L"Далі";
this->IterButton->UseVisualStyleBackColor = true;
this->IterButton->Click += gcnew System::EventHandler(this,
&EmborderingSolution::IterButton_Click);
//
// label11
//
this->label11->AutoSize = true;
this->label11->Location = System::Drawing::Point(589, 15);
this->label11->Name = L"label11";
this->label11->Size = System::Drawing::Size(142, 20);
this->label11->TabIndex = 133;
this->label11->Text = L"Головна матриця";
//
// MainMatrixE
//
this->MainMatrixE->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
this->MainMatrixE->ColumnHeadersVisible = false;
this->MainMatrixE->Location = System::Drawing::Point(470,
```

```
38);
                this->MainMatrixE->Name = L"MainMatrixE";
                this->MainMatrixE->ReadOnly = true;
                this->MainMatrixE->RowHeadersVisible = false;
                this->MainMatrixE->RowHeadersWidth = 62;
                this->MainMatrixE->RowTemplate->Height = 28;
                this->MainMatrixE->Size = System::Drawing::Size(388, 326);
                this->MainMatrixE->TabIndex = 132;
                //
                // menuStrip1
                //
                this->menuStrip1->GripMargin =
System::Windows::Forms::Padding(2, 2, 0, 2);
                this->menuStrip1->ImageScalingSize =
System::Drawing::Size(24, 24);
                this->menuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) { this->вихідToolStripMenuItem });
                this->menuStrip1->Location = System::Drawing::Point(0, 0);
                this->menuStrip1->Name = L"menuStrip1";
                this->menuStrip1->Size = System::Drawing::Size(1707, 33);
                this->menuStrip1->TabIndex = 134;
                this->menuStrip1->Text = L"menuStrip1";
                //
                // вихідToolStripMenuItem
                //
                this->вихідToolStripMenuItem->Name =
L"вихідToolStripMenuItem";
                this->вихідToolStripMenuItem->Size =
System::Drawing::Size(70, 29);
                this->вихідToolStripMenuItem->Text = L"Вихід";
                this->вихідToolStripMenuItem->Click += gcnew
```

```
System::EventHandler(this,
&EmborderingSolution::вихідToolStripMenuItem_Click);
                //
                // EmborderingSolution
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(9, 20);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->AutoScroll = true;
                this->ClientSize = System::Drawing::Size(1707, 1050);
                this->Controls->Add(this->label11);
                this->Controls->Add(this->MainMatrixE);
                this->Controls->Add(this->IterButton);
                this->Controls->Add(this->label10);
                this->Controls->Add(this->AkGrid);
                this->Controls->Add(this->label9);
                this->Controls->Add(this->BView);
                this->Controls->Add(this->label7);
                this->Controls->Add(this->label8);
                this->Controls->Add(this->qView);
                this->Controls->Add(this->rView);
                this->Controls->Add(this->label6);
                this->Controls->Add(this->label5);
                this->Controls->Add(this->AkText);
                this->Controls->Add(this->AkkText);
                this->Controls->Add(this->IterText);
                this->Controls->Add(this->label4);
                this->Controls->Add(this->label3);
                this->Controls->Add(this->VView);
                this->Controls->Add(this->UView);
                this->Controls->Add(this->label2);
```

```
            this->Controls->Add(this->MatrixAView);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->menuStrip1);
            this->MainMenuStrip = this->menuStrip1;
            this->Name = L"EmborderingSolution";
            this->Text = L"EmborderingSolution";
            this->WindowState =
System::Windows::Forms::FormWindowState::Maximized;

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixAView))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>UView))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>VView))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->qView))-
>EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->rView))-
>EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>BView))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>AkGrid))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
```

```
>MainMatrixE))->EndInit();
                this->menuStrip1->ResumeLayout(false);
                this->menuStrip1->PerformLayout();
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
private: void Esolution(Matrix MainMatrix, int iter);
private: Matrix GetMain();
private: System::Void IterButton_Click(System::Object^ sender, System::EventArgs^
e);
private: System::Void вихідToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e);
};
}
```

Файл EmborderingSolution.cpp

```
#include "EmborderingSolution.h"
#include "RevMat.h"

void ReverseMatrix::EmborderingSolution::Esolution(Matrix MainMatrix, int iter)
{
    int Disposal = MainMatrix.GetRow();
    Matrix OldA(1, 1);
    OldA.GetArr()[0][0] = 1 / MainMatrix.GetArr()[0][0];

    if (iter == 0)
    {
        MainMatrixE->RowCount = MainMatrix.GetRow();
        MainMatrixE->ColumnCount = MainMatrix.GetColumn();
```

```
		for (int i = 0; i < MainMatrix.GetRow(); i++)
		{
			for (int j = 0; j < MainMatrix.GetColumn(); j++)
			{
				//Значення матриці
				MainMatrixE->Rows[i]->Cells[j]->Value =
MainMatrix.GetArr()[i][j];
			}
		}
		MainMatrixE-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
		MainMatrixE->AutoResizeColumns();//Стовбці

		IterText->Text = Convert::ToString(1);

		AkGrid->RowCount = 1;
		AkGrid->ColumnCount = 1;

		MatrixAView->RowCount = OldA.GetRow();
		MatrixAView->ColumnCount = OldA.GetColumn();
		for (int i = 0; i < OldA.GetRow(); i++)
		{
			for (int j = 0; j < OldA.GetColumn(); j++)
			{
				//Значення матриці
				MatrixAView->Rows[i]->Cells[j]->Value =
OldA.GetArr()[i][j];
			}
		}
		MatrixAView-
```

```
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
            MatrixAView->AutoResizeColumns();//Стовбці
        }

        for (int k = 1; k < Disposal; k++)
        {
            Matrix A(k + 1, k + 1);
            for (int i = 0; i < k + 1; i++)
            {
                for (int j = 0; j < k + 1; j++)
                {
                    A.GetArr()[i][j] = MainMatrix.GetArr()[i][j];
                }
            }
            if (k == iter)
            {
                MatrixAView->RowCount = A.GetRow();
                MatrixAView->ColumnCount = A.GetColumn();
                for (int i = 0; i < A.GetRow(); i++)
                {
                    for (int j = 0; j < A.GetColumn(); j++)
                    {
                        //Значення матриці
                        MatrixAView->Rows[i]->Cells[j]->Value =
A.GetArr()[i][j];
                    }
                }
                MatrixAView-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
```

```
        MatrixAView->AutoResizeColumns();//Стовбці
}

Matrix U(k, 1);
Matrix V(1, k);
double a = A.GetArr()[k][k];
for (int i = 0; i < k; i++)
{
        U.GetArr()[i][0] = A.GetArr()[i][k];
        V.GetArr()[0][i] = A.GetArr()[k][i];
}
double akk = a - (V * OldA * U).GetArr()[0][0];
Matrix r = MachineZero(OldA * U * (-1 / akk));
Matrix q = MachineZero(V * OldA * (-1 / akk));
Matrix B = MachineZero(OldA - (OldA * U) * q);


for (int i = 0; i < k; i++)
{
        for (int j = 0; j < k; j++)
        {
                A.GetArr()[i][j] = B.GetArr()[i][j];
        }
}
for (int j = 0; j < k; j++)
{
        A.GetArr()[j][k] = r.GetArr()[j][0];
        A.GetArr()[k][j] = q.GetArr()[0][j];
}
A.GetArr()[k][k] = (1 / akk);
```

```
            if (k == iter)
            {
                IterText->Text = Convert::ToString(iter + 1);
                UView->RowCount = U.GetRow();
                UView->ColumnCount = U.GetColumn();
                for (int i = 0; i < U.GetRow(); i++)
                {
                    for (int j = 0; j < U.GetColumn(); j++)
                    {
                        //Значення матриці
                        UView->Rows[i]->Cells[j]->Value =
U.GetArr()[i][j];
                    }
                }
                UView-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
                UView->AutoResizeColumns();//Стовбці

                AkkText->Text = Convert::ToString(a);
                AkText->Text = Convert::ToString(akk);

                VView->RowCount = V.GetRow();
                VView->ColumnCount = V.GetColumn();
                for (int i = 0; i < V.GetRow(); i++)
                {
                    for (int j = 0; j < V.GetColumn(); j++)
                    {
                        //Значення матриці
                        VView->Rows[i]->Cells[j]->Value =
V.GetArr()[i][j];
```

```
            }
        }
        VView->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
        VView->AutoResizeColumns();//Стовбці

        rView->RowCount = r.GetRow();
        rView->ColumnCount = r.GetColumn();
        for (int i = 0; i < r.GetRow(); i++)
        {
            for (int j = 0; j < r.GetColumn(); j++)
            {
                //Значення матриці
                rView->Rows[i]->Cells[j]->Value = r.GetArr()[i][j];
            }
        }
        rView->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
        rView->AutoResizeColumns();//Стовбці

        qView->RowCount = q.GetRow();
        qView->ColumnCount = q.GetColumn();
        for (int i = 0; i < q.GetRow(); i++)
        {
            for (int j = 0; j < q.GetColumn(); j++)
            {
                //Значення матриці
                qView->Rows[i]->Cells[j]->Value = q.GetArr()[i][j];
            }
```

```
                }
                qView->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
                qView->AutoResizeColumns();//Стовбці

                BView->RowCount = B.GetRow();
                BView->ColumnCount = B.GetColumn();
                for (int i = 0; i < B.GetRow(); i++)
                {
                        for (int j = 0; j < B.GetColumn(); j++)
                        {
                                //Значення матриці
                                BView->Rows[i]->Cells[j]->Value = B.GetArr()[i][j];
                        }
                }
                BView->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
                BView->AutoResizeColumns();//Стовбці

                AkGrid->RowCount = A.GetRow();
                AkGrid->ColumnCount = A.GetColumn();
                for (int i = 0; i < A.GetRow(); i++)
                {
                        for (int j = 0; j < A.GetColumn(); j++)
                        {
                                //Значення матриці
                                AkGrid->Rows[i]->Cells[j]->Value = A.GetArr()[i][j];
```

```
                }
            }
            AkGrid->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
            AkGrid->AutoResizeColumns();//Стовбці
        }

        OldA = A;
    }
}

Matrix ReverseMatrix::EmborderingSolution::GetMain()
{
    Matrix SomeMatrix(MainMatrixE->RowCount, MainMatrixE->ColumnCount);

    for (int i = 0; i < MainMatrixE->RowCount; i++)
    {
        for (int j = 0; j < MainMatrixE->ColumnCount; j++)
        {
            SomeMatrix.GetArr()[i][j] = Convert::ToDouble(MainMatrixE->Rows[i]->Cells[j]->Value);
        }
    }

    return SomeMatrix;
}


System::Void
```

```
ReverseMatrix::EmborderingSolution::IterButton_Click(System::Object^ sender,
System::EventArgs^ e)
{
	Matrix MainMatrix = GetMain();
	int iter = Convert::ToInt32(IterText->Text);

	if (iter == MainMatrix.GetRow())
	{
		MessageBox::Show("Це була остання ітерація");
	}
	else
	{
		Esolution(MainMatrix, iter);
	}
	return System::Void();
}

System::Void
ReverseMatrix::EmborderingSolution::вихідToolStripMenuItem_Click(System::Obj
ect^ sender, System::EventArgs^ e)
{
	Application::Exit();
}
```

Файл CellDivisionSolution.h

```
#pragma once
#include "Matrix.h"

namespace ReverseMatrix {

	using namespace System;
```

```
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;

/// <summary>
/// Сводка для CellDivisionSolution
/// </summary>

public ref class CellDivisionSolution : public System::Windows::Forms::Form
{

public:
	CellDivisionSolution(Matrix MainMatrix)
	{
		InitializeComponent();
		//
		//TODO: добавьте код конструктора
		//
		CDResult(MainMatrix);
	}

protected:
	/// <summary>
	/// Освободить все используемые ресурсы.
	/// </summary>
	~CellDivisionSolution()
	{
		if (components)
		{
```

```
                delete components;
            }
        }
private: System::Windows::Forms::Label^ IntroductionLabel;
private: System::Windows::Forms::DataGridView^ MatrixCell11View;
private: System::Windows::Forms::DataGridView^ MatrixCell12View;
private: System::Windows::Forms::DataGridView^ MatrixCell21View;
private: System::Windows::Forms::DataGridView^ MatrixCell22View;




private: System::Windows::Forms::Label^ SecondPhase;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::Label^ label5;
private: System::Windows::Forms::Label^ label6;
private: System::Windows::Forms::Label^ label7;
private: System::Windows::Forms::DataGridView^ R11View;
private: System::Windows::Forms::DataGridView^ R12View;




private: System::Windows::Forms::Label^ label9;
private: System::Windows::Forms::Label^ label10;
private: System::Windows::Forms::DataGridView^ R21View;

private: System::Windows::Forms::Label^ label11;
```

```
        private: System::Windows::Forms::Label^ label12;
        private: System::Windows::Forms::DataGridView^ R22View;

        private: System::Windows::Forms::Label^ label13;
        private: System::Windows::Forms::Label^ label14;
        private: System::Windows::Forms::Label^ label15;
        private: System::Windows::Forms::DataGridView^ ReverseMatrixCD;
        private: System::Windows::Forms::Label^ label8;
        private: System::Windows::Forms::MenuStrip^ menuStrip1;
        private: System::Windows::Forms::ToolStripMenuItem^
вихідToolStripMenuItem;
        private: System::Windows::Forms::TextBox^ RecCount;
        private: System::Windows::Forms::Label^ label16;
        protected:

        protected:

        private:
            /// <summary>
            /// Обязательная переменная конструктора.
            /// </summary>
            System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Требуемый метод для поддержки конструктора — не изменяйте
            /// содержимое этого метода с помощью редактора кода.
            /// </summary>
            void InitializeComponent(void)
            {
                this->IntroductionLabel = (gcnew
```

```
System::Windows::Forms::Label());
                this->MatrixCell11View = (gcnew
System::Windows::Forms::DataGridView());
                this->MatrixCell12View = (gcnew
System::Windows::Forms::DataGridView());
                this->MatrixCell21View = (gcnew
System::Windows::Forms::DataGridView());
                this->MatrixCell22View = (gcnew
System::Windows::Forms::DataGridView());
                this->SecondPhase = (gcnew System::Windows::Forms::Label());
                this->label1 = (gcnew System::Windows::Forms::Label());
                this->label2 = (gcnew System::Windows::Forms::Label());
                this->label3 = (gcnew System::Windows::Forms::Label());
                this->label4 = (gcnew System::Windows::Forms::Label());
                this->label5 = (gcnew System::Windows::Forms::Label());
                this->label6 = (gcnew System::Windows::Forms::Label());
                this->label7 = (gcnew System::Windows::Forms::Label());
                this->R11View = (gcnew
System::Windows::Forms::DataGridView());
                this->R12View = (gcnew
System::Windows::Forms::DataGridView());
                this->label9 = (gcnew System::Windows::Forms::Label());
                this->label10 = (gcnew System::Windows::Forms::Label());
                this->R21View = (gcnew
System::Windows::Forms::DataGridView());
                this->label11 = (gcnew System::Windows::Forms::Label());
                this->label12 = (gcnew System::Windows::Forms::Label());
                this->R22View = (gcnew
System::Windows::Forms::DataGridView());
                this->label13 = (gcnew System::Windows::Forms::Label());
                this->label14 = (gcnew System::Windows::Forms::Label());
```

```
            this->label15 = (gcnew System::Windows::Forms::Label());
            this->ReverseMatrixCD = (gcnew
System::Windows::Forms::DataGridView());
            this->label8 = (gcnew System::Windows::Forms::Label());
            this->menuStrip1 = (gcnew
System::Windows::Forms::MenuStrip());
            this->вихідToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->RecCount = (gcnew System::Windows::Forms::TextBox());
            this->label16 = (gcnew System::Windows::Forms::Label());

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixCell11View))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixCell12View))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixCell21View))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixCell22View))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>R11View))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>R12View))->BeginInit();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>R21View))->BeginInit();
```

```
(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->R22View))->BeginInit();

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->ReverseMatrixCD))->BeginInit();
                this->menuStrip1->SuspendLayout();
                this->SuspendLayout();
                //
                // IntroductionLabel
                //
                this->IntroductionLabel->AutoSize = true;
                this->IntroductionLabel->Location = System::Drawing::Point(511, 9);
                this->IntroductionLabel->Name = L"IntroductionLabel";
                this->IntroductionLabel->Size = System::Drawing::Size(482, 20);
                this->IntroductionLabel->TabIndex = 0;
                this->IntroductionLabel->Text = L"Спочатку розіб\'ємо основну матрицю на 4 підматриці (клітини)";
                //
                // MatrixCell11View
                //
                this->MatrixCell11View->AllowUserToAddRows = false;
                this->MatrixCell11View->AllowUserToDeleteRows = false;
                this->MatrixCell11View->AllowUserToResizeColumns = false;
                this->MatrixCell11View->AllowUserToResizeRows = false;
                this->MatrixCell11View->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                this->MatrixCell11View->ColumnHeadersVisible = false;
                this->MatrixCell11View->Location =
```

```
System::Drawing::Point(105, 80);
                this->MatrixCell11View->Name = L"MatrixCell11View";
                this->MatrixCell11View->ReadOnly = true;
                this->MatrixCell11View->RowHeadersVisible = false;
                this->MatrixCell11View->RowHeadersWidth = 62;
                this->MatrixCell11View->RowTemplate->Height = 28;
                this->MatrixCell11View->Size = System::Drawing::Size(267,
189);
                this->MatrixCell11View->TabIndex = 1;
                //
                // MatrixCell12View
                //
                this->MatrixCell12View->AllowUserToAddRows = false;
                this->MatrixCell12View->AllowUserToDeleteRows = false;
                this->MatrixCell12View->AllowUserToResizeColumns = false;
                this->MatrixCell12View->AllowUserToResizeRows = false;
                this->MatrixCell12View->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
                this->MatrixCell12View->ColumnHeadersVisible = false;
                this->MatrixCell12View->Location =
System::Drawing::Point(430, 80);
                this->MatrixCell12View->Name = L"MatrixCell12View";
                this->MatrixCell12View->ReadOnly = true;
                this->MatrixCell12View->RowHeadersVisible = false;
                this->MatrixCell12View->RowHeadersWidth = 62;
                this->MatrixCell12View->RowTemplate->Height = 28;
                this->MatrixCell12View->Size = System::Drawing::Size(267,
189);
                this->MatrixCell12View->TabIndex = 2;
                //
```

```
            // MatrixCell21View
            //
            this->MatrixCell21View->AllowUserToAddRows = false;
            this->MatrixCell21View->AllowUserToDeleteRows = false;
            this->MatrixCell21View->AllowUserToResizeColumns = false;
            this->MatrixCell21View->AllowUserToResizeRows = false;
            this->MatrixCell21View->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->MatrixCell21View->ColumnHeadersVisible = false;
            this->MatrixCell21View->Location =
System::Drawing::Point(752, 80);
            this->MatrixCell21View->Name = L"MatrixCell21View";
            this->MatrixCell21View->ReadOnly = true;
            this->MatrixCell21View->RowHeadersVisible = false;
            this->MatrixCell21View->RowHeadersWidth = 62;
            this->MatrixCell21View->RowTemplate->Height = 28;
            this->MatrixCell21View->Size = System::Drawing::Size(267,
189);
            this->MatrixCell21View->TabIndex = 3;
            //
            // MatrixCell22View
            //
            this->MatrixCell22View->AllowUserToAddRows = false;
            this->MatrixCell22View->AllowUserToDeleteRows = false;
            this->MatrixCell22View->AllowUserToResizeColumns = false;
            this->MatrixCell22View->AllowUserToResizeRows = false;
            this->MatrixCell22View->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->MatrixCell22View->ColumnHeadersVisible = false;
```

```
            this->MatrixCell22View->Location =
System::Drawing::Point(1074, 80);
            this->MatrixCell22View->Name = L"MatrixCell22View";
            this->MatrixCell22View->ReadOnly = true;
            this->MatrixCell22View->RowHeadersVisible = false;
            this->MatrixCell22View->RowHeadersWidth = 62;
            this->MatrixCell22View->RowTemplate->Height = 28;
            this->MatrixCell22View->Size = System::Drawing::Size(267,
189);
            this->MatrixCell22View->TabIndex = 4;
            //
            // SecondPhase
            //
            this->SecondPhase->AutoSize = true;
            this->SecondPhase->Location = System::Drawing::Point(511,
296);
            this->SecondPhase->Name = L"SecondPhase";
            this->SecondPhase->Size = System::Drawing::Size(463, 20);
            this->SecondPhase->TabIndex = 5;
            this->SecondPhase->Text = L"Далі знаходимо клітини до
оберненої матриці по формулам";
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Location = System::Drawing::Point(226, 54);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(38, 20);
            this->label1->TabIndex = 6;
            this->label1->Text = L"A11";
            //
```

```
// label2
//
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(539, 54);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(38, 20);
this->label2->TabIndex = 7;
this->label2->Text = L"A12";
//
// label3
//
this->label3->AutoSize = true;
this->label3->Location = System::Drawing::Point(1184, 57);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(38, 20);
this->label3->TabIndex = 8;
this->label3->Text = L"A22";
//
// label4
//
this->label4->AutoSize = true;
this->label4->Location = System::Drawing::Point(869, 57);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(38, 20);
this->label4->TabIndex = 9;
this->label4->Text = L"A21";
//
// label5
//
this->label5->AutoSize = true;
this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft
```

```
Sans Serif", 14));
            this->label5->Location = System::Drawing::Point(12, 414);
            this->label5->Name = L"label5";
            this->label5->Size = System::Drawing::Size(386, 32);
            this->label5->TabIndex = 10;
            this->label5->Text = L"R11 = (A11 - A12 A22 A21) = ";
            //
            // label6
            //
            this->label6->AutoSize = true;
            this->label6->Location = System::Drawing::Point(230, 394);
            this->label6->Name = L"label6";
            this->label6->Size = System::Drawing::Size(23, 20);
            this->label6->TabIndex = 11;
            this->label6->Text = L"-1";
            //
            // label7
            //
            this->label7->AutoSize = true;
            this->label7->Location = System::Drawing::Point(349, 394);
            this->label7->Name = L"label7";
            this->label7->Size = System::Drawing::Size(23, 20);
            this->label7->TabIndex = 12;
            this->label7->Text = L"-1";
            //
            // R11View
            //
            this->R11View->AllowUserToAddRows = false;
            this->R11View->AllowUserToDeleteRows = false;
            this->R11View->AllowUserToResizeColumns = false;
            this->R11View->AllowUserToResizeRows = false;
```

```
            this->R11View->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->R11View->ColumnHeadersVisible = false;
            this->R11View->Location = System::Drawing::Point(395, 344);
            this->R11View->Name = L"R11View";
            this->R11View->ReadOnly = true;
            this->R11View->RowHeadersVisible = false;
            this->R11View->RowHeadersWidth = 62;
            this->R11View->RowTemplate->Height = 28;
            this->R11View->Size = System::Drawing::Size(267, 189);
            this->R11View->TabIndex = 13;
            //
            // R12View
            //
            this->R12View->AllowUserToAddRows = false;
            this->R12View->AllowUserToDeleteRows = false;
            this->R12View->AllowUserToResizeColumns = false;
            this->R12View->AllowUserToResizeRows = false;
            this->R12View->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->R12View->ColumnHeadersVisible = false;
            this->R12View->Location = System::Drawing::Point(1074, 344);
            this->R12View->Name = L"R12View";
            this->R12View->ReadOnly = true;
            this->R12View->RowHeadersVisible = false;
            this->R12View->RowHeadersWidth = 62;
            this->R12View->RowTemplate->Height = 28;
            this->R12View->Size = System::Drawing::Size(267, 189);
            this->R12View->TabIndex = 17;
```

```
            //
            // label9
            //
            this->label9->AutoSize = true;
            this->label9->Location = System::Drawing::Point(992, 394);
            this->label9->Name = L"label9";
            this->label9->Size = System::Drawing::Size(23, 20);
            this->label9->TabIndex = 15;
            this->label9->Text = L"-1";
            //
            // label10
            //
            this->label10->AutoSize = true;
            this->label10->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14));
            this->label10->Location = System::Drawing::Point(739, 414);
            this->label10->Name = L"label10";
            this->label10->Size = System::Drawing::Size(297, 32);
            this->label10->TabIndex = 14;
            this->label10->Text = L"R12 = -R11 A12 A22 =";
            //
            // R21View
            //
            this->R21View->AllowUserToAddRows = false;
            this->R21View->AllowUserToDeleteRows = false;
            this->R21View->AllowUserToResizeColumns = false;
            this->R21View->AllowUserToResizeRows = false;
            this->R21View->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
            this->R21View->ColumnHeadersVisible = false;
```

```
this->R21View->Location = System::Drawing::Point(395, 565);
this->R21View->Name = L"R21View";
this->R21View->ReadOnly = true;
this->R21View->RowHeadersVisible = false;
this->R21View->RowHeadersWidth = 62;
this->R21View->RowTemplate->Height = 28;
this->R21View->Size = System::Drawing::Size(267, 189);
this->R21View->TabIndex = 21;
//
// label11
//
this->label11->AutoSize = true;
this->label11->Location = System::Drawing::Point(217, 615);
this->label11->Name = L"label11";
this->label11->Size = System::Drawing::Size(23, 20);
this->label11->TabIndex = 19;
this->label11->Text = L"-1";
//
// label12
//
this->label12->AutoSize = true;
this->label12->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14));
this->label12->Location = System::Drawing::Point(75, 635);
this->label12->Name = L"label12";
this->label12->Size = System::Drawing::Size(297, 32);
this->label12->TabIndex = 18;
this->label12->Text = L"R21 = -A22 A21 R21 =";
//
// R22View
//
```

```
this->R22View->AllowUserToAddRows = false;
this->R22View->AllowUserToDeleteRows = false;
this->R22View->AllowUserToResizeColumns = false;
this->R22View->AllowUserToResizeRows = false;
this->R22View->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
this->R22View->ColumnHeadersVisible = false;
this->R22View->Location = System::Drawing::Point(1074, 582);
this->R22View->Name = L"R22View";
this->R22View->ReadOnly = true;
this->R22View->RowHeadersVisible = false;
this->R22View->RowHeadersWidth = 62;
this->R22View->RowTemplate->Height = 28;
this->R22View->Size = System::Drawing::Size(267, 189);
this->R22View->TabIndex = 25;
//
// label13
//
this->label13->AutoSize = true;
this->label13->Location = System::Drawing::Point(907, 615);
this->label13->Name = L"label13";
this->label13->Size = System::Drawing::Size(23, 20);
this->label13->TabIndex = 24;
this->label13->Text = L"-1";
//
// label14
//
this->label14->AutoSize = true;
this->label14->Location = System::Drawing::Point(828, 615);
this->label14->Name = L"label14";
```

```
this->label14->Size = System::Drawing::Size(23, 20);
this->label14->TabIndex = 23;
this->label14->Text = L"-1";
//
// label15
//
this->label15->AutoSize = true;
this->label15->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14));
this->label15->Location = System::Drawing::Point(700, 635);
this->label15->Name = L"label15";
this->label15->Size = System::Drawing::Size(368, 32);
this->label15->TabIndex = 22;
this->label15->Text = L"R22 = A22 - A22 A21 B12 = ";
//
// ReverseMatrixCD
//
this->ReverseMatrixCD->AllowUserToAddRows = false;
this->ReverseMatrixCD->AllowUserToDeleteRows = false;
this->ReverseMatrixCD->AllowUserToResizeColumns = false;
this->ReverseMatrixCD->AllowUserToResizeRows = false;
this->ReverseMatrixCD->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSiz
e;
this->ReverseMatrixCD->ColumnHeadersVisible = false;
this->ReverseMatrixCD->Location =
System::Drawing::Point(558, 769);
this->ReverseMatrixCD->Name = L"ReverseMatrixCD";
this->ReverseMatrixCD->ReadOnly = true;
this->ReverseMatrixCD->RowHeadersVisible = false;
this->ReverseMatrixCD->RowHeadersWidth = 62;
```

```
this->ReverseMatrixCD->RowTemplate->Height = 28;
this->ReverseMatrixCD->Size = System::Drawing::Size(372,
269);
this->ReverseMatrixCD->TabIndex = 26;
//
// label8
//
this->label8->AutoSize = true;
this->label8->Location = System::Drawing::Point(113, 889);
this->label8->Name = L"label8";
this->label8->Size = System::Drawing::Size(401, 20);
this->label8->TabIndex = 27;
this->label8->Text = L"Збираємо клітини та отримуємо
обернену матрицю";
//
// menuStrip1
//
this->menuStrip1->GripMargin =
System::Windows::Forms::Padding(2, 2, 0, 2);
this->menuStrip1->ImageScalingSize =
System::Drawing::Size(24, 24);
this->menuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) { this->вихідToolStripMenuItem });
this->menuStrip1->Location = System::Drawing::Point(0, 0);
this->menuStrip1->Name = L"menuStrip1";
this->menuStrip1->Size = System::Drawing::Size(1395, 33);
this->menuStrip1->TabIndex = 28;
this->menuStrip1->Text = L"menuStrip1";
//
// вихідToolStripMenuItem
//
```

```
            this->вихідToolStripMenuItem->Name =
L"вихідToolStripMenuItem";
            this->вихідToolStripMenuItem->Size =
System::Drawing::Size(70, 29);
            this->вихідToolStripMenuItem->Text = L"Вихід";
            this->вихідToolStripMenuItem->Click += gcnew
System::EventHandler(this,
&CellDivisionSolution::вихідToolStripMenuItem_Click);
            //
            // RecCount
            //
            this->RecCount->Location = System::Drawing::Point(1036, 930);
            this->RecCount->Name = L"RecCount";
            this->RecCount->ReadOnly = true;
            this->RecCount->Size = System::Drawing::Size(100, 26);
            this->RecCount->TabIndex = 29;
            //
            // label16
            //
            this->label16->AutoSize = true;
            this->label16->Location = System::Drawing::Point(956, 889);
            this->label16->Name = L"label16";
            this->label16->Size = System::Drawing::Size(253, 20);
            this->label16->TabIndex = 30;
            this->label16->Text = L"Кількість рекурсивних входжень";
            //
            // CellDivisionSolution
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(9, 20);
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
```

```
this->AutoScroll = true;
this->ClientSize = System::Drawing::Size(1395, 1050);
this->Controls->Add(this->label16);
this->Controls->Add(this->RecCount);
this->Controls->Add(this->label8);
this->Controls->Add(this->ReverseMatrixCD);
this->Controls->Add(this->R22View);
this->Controls->Add(this->label13);
this->Controls->Add(this->label14);
this->Controls->Add(this->label15);
this->Controls->Add(this->R21View);
this->Controls->Add(this->label11);
this->Controls->Add(this->label12);
this->Controls->Add(this->R12View);
this->Controls->Add(this->label9);
this->Controls->Add(this->label10);
this->Controls->Add(this->R11View);
this->Controls->Add(this->label7);
this->Controls->Add(this->label6);
this->Controls->Add(this->label5);
this->Controls->Add(this->label4);
this->Controls->Add(this->label3);
this->Controls->Add(this->label2);
this->Controls->Add(this->label1);
this->Controls->Add(this->SecondPhase);
this->Controls->Add(this->MatrixCell22View);
this->Controls->Add(this->MatrixCell21View);
this->Controls->Add(this->MatrixCell12View);
this->Controls->Add(this->MatrixCell11View);
this->Controls->Add(this->IntroductionLabel);
this->Controls->Add(this->menuStrip1);
```

```
            this->MainMenuStrip = this->menuStrip1;
            this->Name = L"CellDivisionSolution";
            this->Text = L"CellDivisionSolution";
            this->WindowState =
System::Windows::Forms::FormWindowState::Maximized;

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixCell11View))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixCell12View))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixCell21View))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>MatrixCell22View))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>R11View))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>R12View))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>R21View))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>R22View))->EndInit();

    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
```

```
>ReverseMatrixCD))->EndInit();
                this->menuStrip1->ResumeLayout(false);
                this->menuStrip1->PerformLayout();
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion


private: System::Void вихідToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e);
private: void CDResult(Matrix MainMatrix);
};
}
```

Файл CellDivisionSolution.cpp

```
#include "CellDivisionSolution.h"
#include "RevMat.h"

System::Void
ReverseMatrix::CellDivisionSolution::вихідToolStripMenuItem_Click(System::Obje
ct^ sender, System::EventArgs^ e)
{
    Application::Exit();
    return System::Void();
}

void ReverseMatrix::CellDivisionSolution::CDResult(Matrix MainMatrix)
{
    int count = 0;
    bool ZeroDiv = false;
```

```
        Matrix MatrixCell11 = CellDivisionInitialization(MainMatrix, 1);
        Matrix MatrixCell12 = CellDivisionInitialization(MainMatrix, 2);
        Matrix MatrixCell21 = CellDivisionInitialization(MainMatrix, 3);
        Matrix MatrixCell22 = CellDivisionInitialization(MainMatrix, 4);
        count++;
        Matrix MatrixCell22Inverse = CellDivision(MatrixCell22, &count,
&ZeroDiv);

        count++;
        Matrix MatrixR11 = MachineZero(CellDivision(MatrixCell11 - (MatrixCell12
* (MatrixCell22Inverse * MatrixCell21)), &count, &ZeroDiv));
        Matrix MatrixR12 = MachineZero(((MatrixR11 * (-1)) * MatrixCell12) *
MatrixCell22Inverse);
        Matrix MatrixR21 = MachineZero((MatrixCell22Inverse * (-1)) *
MatrixCell21 * MatrixR11);
        Matrix MatrixR22 = MachineZero(MatrixCell22Inverse -
(MatrixCell22Inverse * MatrixCell21 * MatrixR12));

        Matrix ReverseMatrix = CellDivisionBuild(MainMatrix, MatrixR11,
MatrixR12, MatrixR21, MatrixR22);

        RecCount->Text = Convert::ToString(count);
        MatrixCell11View->RowCount = MatrixCell11.GetRow();
        MatrixCell11View->ColumnCount = MatrixCell11.GetColumn();
        for (int i = 0; i < MatrixCell11.GetRow(); i++)
        {
                for (int j = 0; j < MatrixCell11.GetColumn(); j++)
                {
                        //Назва таблиці у лівому кутку
                        MatrixCell11View->TopLeftHeaderCell->Value = "A11";
                        //Номери столбців
```

```
                MatrixCell11View->Columns[j]->HeaderCell->Value =
Convert::ToString(j + 1);
                //Номери рядків
                MatrixCell11View->Rows[i]->HeaderCell->Value =
Convert::ToString(i + 1);
                //Значення матриці
                MatrixCell11View->Rows[i]->Cells[j]->Value =
MatrixCell11.GetArr()[i][j];
            }
        }
        MatrixCell11View-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
        MatrixCell11View->AutoResizeColumns();//Стовбці


        MatrixCell12View->RowCount = MatrixCell12.GetRow();
        MatrixCell12View->ColumnCount = MatrixCell12.GetColumn();
        for (int i = 0; i < MatrixCell12.GetRow(); i++)
        {
            for (int j = 0; j < MatrixCell12.GetColumn(); j++)
            {
                //Назва таблиці у лівому кутку
                MatrixCell12View->TopLeftHeaderCell->Value = "A12";
                //Номери столбців
                MatrixCell12View->Columns[j]->HeaderCell->Value =
Convert::ToString(j + 1);
                //Номери рядків
                MatrixCell12View->Rows[i]->HeaderCell->Value =
Convert::ToString(i + 1);
                //Значення матриці
```

```
                MatrixCell12View->Rows[i]->Cells[j]->Value =
MatrixCell12.GetArr()[i][j];
            }
        }
        MatrixCell12View-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
        MatrixCell12View->AutoResizeColumns();//Стовбці

        MatrixCell21View->RowCount = MatrixCell21.GetRow();
        MatrixCell21View->ColumnCount = MatrixCell21.GetColumn();
        for (int i = 0; i < MatrixCell21.GetRow(); i++)
        {
            for (int j = 0; j < MatrixCell21.GetColumn(); j++)
            {
                //Назва таблиці у лівому кутку
                MatrixCell21View->TopLeftHeaderCell->Value = "A21";
                //Номери столбців
                MatrixCell21View->Columns[j]->HeaderCell->Value =
Convert::ToString(j + 1);
                //Номери рядків
                MatrixCell21View->Rows[i]->HeaderCell->Value =
Convert::ToString(i + 1);
                //Значення матриці
                MatrixCell21View->Rows[i]->Cells[j]->Value =
MatrixCell21.GetArr()[i][j];
            }
        }
        MatrixCell21View-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
```

```
	MatrixCell21View->AutoResizeColumns();//Стовбці

	MatrixCell22View->RowCount = MatrixCell22.GetRow();
	MatrixCell22View->ColumnCount = MatrixCell22.GetColumn();
	for (int i = 0; i < MatrixCell22.GetRow(); i++)
	{
		for (int j = 0; j < MatrixCell22.GetColumn(); j++)
		{
			//Назва таблиці у лівому кутку
			MatrixCell22View->TopLeftHeaderCell->Value = "A22";
			//Номери столбців
			MatrixCell22View->Columns[j]->HeaderCell->Value =
Convert::ToString(j + 1);
			//Номери рядків
			MatrixCell22View->Rows[i]->HeaderCell->Value =
Convert::ToString(i + 1);
			//Значення матриці
			MatrixCell22View->Rows[i]->Cells[j]->Value =
MatrixCell22.GetArr()[i][j];
		}
	}
	MatrixCell22View-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
	MatrixCell22View->AutoResizeColumns();//Стовбці

	R11View->RowCount = MatrixR11.GetRow();
	R11View->ColumnCount = MatrixR11.GetColumn();
	for (int i = 0; i < MatrixR11.GetRow(); i++)
	{
		for (int j = 0; j < MatrixR11.GetColumn(); j++)
```

```
            {
                //Назва таблиці у лівому кутку
                R11View->TopLeftHeaderCell->Value = "R11";
                //Номери столбців
                R11View->Columns[j]->HeaderCell->Value =
Convert::ToString(j + 1);
                //Номери рядків
                R11View->Rows[i]->HeaderCell->Value = Convert::ToString(i +
1);
                //Значення матриці
                R11View->Rows[i]->Cells[j]->Value = MatrixR11.GetArr()[i][j];
            }
        }
        R11View-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
        R11View->AutoResizeColumns();//Стовбці

        R12View->RowCount = MatrixR12.GetRow();
        R12View->ColumnCount = MatrixR12.GetColumn();
        for (int i = 0; i < MatrixR12.GetRow(); i++)
        {
            for (int j = 0; j < MatrixR12.GetColumn(); j++)
            {
                //Назва таблиці у лівому кутку
                R12View->TopLeftHeaderCell->Value = "R12";
                //Номери столбців
                R12View->Columns[j]->HeaderCell->Value =
Convert::ToString(j + 1);
                //Номери рядків
                R12View->Rows[i]->HeaderCell->Value = Convert::ToString(i +
```

```
1);
                    //Значення матриці
                    R12View->Rows[i]->Cells[j]->Value = MatrixR12.GetArr()[i][j];
                }
            }
            R12View->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
            R12View->AutoResizeColumns();//Стовбці

            R21View->RowCount = MatrixR21.GetRow();
            R21View->ColumnCount = MatrixR21.GetColumn();
            for (int i = 0; i < MatrixR21.GetRow(); i++)
            {
                for (int j = 0; j < MatrixR21.GetColumn(); j++)
                {
                    //Назва таблиці у лівому кутку
                    R21View->TopLeftHeaderCell->Value = "R21";
                    //Номери столбців
                    R21View->Columns[j]->HeaderCell->Value = Convert::ToString(j + 1);
                    //Номери рядків
                    R21View->Rows[i]->HeaderCell->Value = Convert::ToString(i + 1);
                    //Значення матриці
                    R21View->Rows[i]->Cells[j]->Value = MatrixR21.GetArr()[i][j];
                }
            }
            R21View->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);
```

```
        R21View->AutoResizeColumns();//Стовбці

        R22View->RowCount = MatrixR22.GetRow();
        R22View->ColumnCount = MatrixR22.GetColumn();
        for (int i = 0; i < MatrixR22.GetRow(); i++)
        {
            for (int j = 0; j < MatrixR22.GetColumn(); j++)
            {
                //Назва таблиці у лівому кутку
                R22View->TopLeftHeaderCell->Value = "R22";
                //Номери столбців
                R22View->Columns[j]->HeaderCell->Value =
Convert::ToString(j + 1);
                //Номери рядків
                R22View->Rows[i]->HeaderCell->Value = Convert::ToString(i +
1);

                //Значення матриці
                R22View->Rows[i]->Cells[j]->Value = MatrixR22.GetArr()[i][j];
            }
        }
        R22View-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
        R22View->AutoResizeColumns();//Стовбці

        ReverseMatrixCD->RowCount = ReverseMatrix.GetRow();
        ReverseMatrixCD->ColumnCount = ReverseMatrix.GetColumn();
        for (int i = 0; i < ReverseMatrix.GetRow(); i++)
        {
            for (int j = 0; j < ReverseMatrix.GetColumn(); j++)
            {
```

```
                    //Назва таблиці у лівому кутку
                    ReverseMatrixCD->TopLeftHeaderCell->Value = "Обернена
матриця";
                    //Номери столбців
                    ReverseMatrixCD->Columns[j]->HeaderCell->Value =
Convert::ToString(j + 1);
                    //Номери рядків
                    ReverseMatrixCD->Rows[i]->HeaderCell->Value =
Convert::ToString(i + 1);
                    //Значення матриці
                    ReverseMatrixCD->Rows[i]->Cells[j]->Value =
ReverseMatrix.GetArr()[i][j];
                }
        }
        ReverseMatrixCD-
>AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSi
zeToAllHeaders);
        ReverseMatrixCD->AutoResizeColumns();//Стовбці
}
```